FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2018/2019

Trabajo de Fin de Máster

TITULO: APRENDIZAJE AUTOMATICO PARA CLASIFICAR DESCUENTOS EN PRENDAS DE ROPA

***Alumno*: Belén Rodríguez Cánovas**

***Tutor*: Javier Portela**

Septiembre de 2019

UNIVERSIDAD COMPLUTENSE
MADRID

Debido a la confidencialidad del conjunto de datos la divulgación total o parcial de este documento queda sujeta a la previa autorización.

Este trabajo no hubiera sido posible sin la gentil cesión de los datos por la compañía StyleSage con la que se ha firmado un acuerdo de no divulgación. .

Doy las gracias a la Facultad de Estadística de la Universidad Complutense y a todos sus excelentes profesores y profesoras que han despertado mi interés por la estadística en el grado y posteriormente en el máster.

Contenido

PARTE I: MARCO TEORICO

CAPITULO I: INTERES, OBJETIVOS Y ESTRUCTURA DE LA INVESTIGACION

1.1. Interés del tema

El presente trabajo reúne tres fenómenos que están de moda en la actualidad. Por un lado, y valga la redundancia, la industria de la moda, y por otro el *Machine Learning* o aprendizaje automático junto con Big Data.

Desde hace ya tiempo la industria de la moda ha atraído el interés de académicos y profesionales del sector. Su estudio ha sido aproximado desde distintas disciplinas y muchos trabajos se han centrado en la psicología y sociología de la moda así como describir el proceso de adopción de la moda en la población (p.ej. Wills y Midgley, 1973). Dada su importancia en la economía también es un fenómeno de gran interés en la literatura sobre organización, dirección y gestión (Cappetta y Gioia, 2005; Djelic y Ainamo, 1999; Richardson, 1996). El mercado de la moda se caracteriza por sus rápidos cambios y muestra las siguientes características: ciclos de vida cortos, alta volatilidad, baja predecibilidad y compra de alto impulso (Christopher, M. et al.; 2004).

De forma general, el aprendizaje automático (*Machine Learning*) es la ciencia de los algoritmos que permite comprender y obtener valor de los datos más allá de lo aparentemente visible. Se refiere a aprendizaje porque los algoritmos que se procesan facilitan la búsqueda de patrones en los datos para poder realizar con el conjunto de datos predicciones o clasificaciones. Gracias a la estadística y la probabilidad las máquinas son “capaces” de aprender manejando información que el ser humano no podría procesar.

Este trabajo ha sido posible gracias a una fuente de datos proporcionada por la *start-up* StyleSage tecnológica del área de Big Data. Esta fuente es real, fiable y contiene datos masivos de la industria de la moda que permiten nutrir varios algoritmos con el fin de extraer información que contenga valor para las empresas de la industria. En concreto, el conjunto de datos que se va a tratar está orientado al estudio de los descuentos en precio de vestidos de mujer. Dispone de 432.000 casos y no puede ser descargado libremente, de ahí la confidencialidad de la estructura de la base de datos y partes de este trabajo.

Para el almacenamiento y procesamiento de los datos se utiliza la biblioteca “pandas” orientada a Python gracias a la cual se puede trabajar de forma flexible con datos. Se desarrolla un programa en Python y se utilizan distintas bibliotecas siendo una de ellas la de libre uso *scikit-learn* (*sklearn*) que contiene numerosos algoritmos de clasificación. Estos algoritmos han sido utilizados previamente y cada uno tiene un comportamiento distinto. Dependiendo del enfoque, los algoritmos se pueden evaluar desde distintas medidas que abarcan desde la precisión a la hora de clasificar datos o el consumo de recursos de la máquina (memoria del sistema, tiempo de ajuste y de predicción) no siendo esto último un objetivo de estudio en este trabajo. Lo que se

busca es determinar qué algoritmo es el más adecuado para dar respuesta al objetivo planteado en la investigación.

1.2. Objetivos de la investigación

Este estudio persigue fundamentalmente dos objetivos. Por un lado, el objetivo principal es, a partir de un conjunto de datos sobre atributos de prendas de ropa, en concreto, vestidos de mujer, tratar de clasificar los porcentajes de descuento en precio aplicados a la prenda. Esta información puede ser valiosa para la toma de decisiones sobre la política de descuento de precios en el surtido de productos de empresas del sector de la moda y provee información sobre las estrategias de los competidores. A su vez, esta información puede ser usada en otro algoritmo junto a otras variables para tomar otras decisiones. Por otro lado, como objetivo secundario se pretende mostrar todo el proceso de aplicación de técnicas de aprendizaje automático en un conjunto de datos de acceso no libre y de naturaleza compleja usando las librerías de Python. No es muy frecuente tener la oportunidad de obtener datos reales que no estén publicados y ya hayan sido procesados y entrenados previamente.

Dado que el objetivo primario es clasificar a partir de una muestra de entrenamiento etiquetada se trata claramente de un caso de aprendizaje supervisado (cada caso viene con el *output* esperado). Cabe señalar que no hay un flujo de datos continuamente entrando en el sistema porque no se realiza en tiempo real en la empresa y tampoco hay una necesidad particular de ajustarse a datos que cambian rápidamente siendo los datos lo suficientemente adecuados para no necesitar programación paralela por lo que el aprendizaje *plain batch* es adecuado.

1.3. Estructura del trabajo

Para exponer este trabajo el presente documento se ha organizado en una parte teórica y otra empírica además de este primer capítulo que comprende la introducción, el objetivo y la estructura. La parte primera muestra el marco teórico relacionado con el tema de interés y consiste en cuatro breves capítulos. El capítulo segundo con cinco apartados presenta el concepto del fenómeno moda, su origen, los tipos que existen así como unos datos de la industria y una descripción sobre la naturaleza del negocio de la moda. El capítulo tercero tiene dos apartados. Uno se dedica a describir el fenómeno de Big Data, su origen y concepto mientras que el segundo apartado hace referencia a los trabajos existentes en la literatura académica sobre moda y Big Data. El capítulo cuarto ahonda en el Aprendizaje Automático o *Machine Learning* con seis apartados que tratan el concepto, su origen, el aprendizaje automático supervisado, los tipos de algoritmos existentes. Los capítulos cinco, seis, siete describen toda la parte empírica: el conjunto de datos, el procesado, el entrenamiento de los algoritmos y los resultados. Finalmente se describen conclusiones y limitaciones en el capítulo octavo. Se añade la bibliografía empleada y los anexos necesarios al final del documento.

CAPITULO II: SOBRE LA INDUSTRIA DE LA MODA

El objetivo del presente capítulo es ofrecer un conocimiento sobre cómo se aproxima el fenómeno moda o *fashion* en el mundo académico así como en el empresarial. Para ello se ha estructurado en un total de seis epígrafes. El primero presenta el concepto de moda. El segundo epígrafe hace una referencia muy breve al origen y evolución de la moda como industria. El tercero describe los distintos tipos de moda que se pueden identificar en el mercado. Seguidamente, el cuarto epígrafe comparte las últimas tendencias de mercado de la moda para ilustrar la importancia que tiene este sector y porqué resulta un tema de gran interés. El quinto apartado hace referencia a la naturaleza del fenómeno moda. El último apartado finalmente presenta una descripción del modelo de negocio de la moda que resulta de utilidad para la interpretación del conjunto de datos con el que se realiza la investigación empírica.

2.1. Concepto de moda o fashion

El fenómeno de la moda está ampliamente documentado tanto en el mundo académico como empresarial y abundan las definiciones sobre moda. Ya en el siglo XIX los primeros intelectuales mostraron interés por la moda como Balzac en el Tratado de la vida elegante de 1949. Según el *Barnhart Dictionary of Etymology*, de 1988, el origen del significado de moda se remonta al año 1300 cuando ya se hacía referencia al estilo y las formas.

Etimológicamente, las palabras moda, *mode* en francés y *fashion* en inglés, derivan del latín pero de distintos vocablos, de ahí a que tengan un significado distinto. En español y francés, moda y *mode* derivan de *mos (modus, modo y medida*), significando: a) Costumbre, usanza, hábito, tradición; b) Ley, regla, norma y; c) Buenas costumbres, moralidad. En inglés, *fashion* deriva del vocablo latino *factio* que significa hacer, dar forma o aparentar proveniente del francés antiguo *façon (Oxford Dictionaries, 2018).* Por lo tanto, el uso original estaba vinculado con la realización de actividades, era algo que alguien había hecho (Malcom, 1996), mientras que en la actualidad, *fashion* se refiera a algo que alguien lleva puesto. A principios del siglo XVI no se decía “estar pasado de moda” sino “fuera de forma”[1]. Es a mediados del siglo XVI cuando *fashion* toma el sentido de estilo popular de ropa o forma de comportarse.

Existen muchas definiciones sobre la moda y el objetivo de este trabajo no es revisar todas y cada una de ellas. Se presentan a título informativo algunas de obligada referencia. Según el Diccionario de la Real Academia Española el término moda (del francés *mode*, y del latín modus, "modo" o "medida"), se remonta al siglo XV y significa “uso, modo o costumbre y que está en boga durante algún tiempo, o en determinado país” (*RAE, 2018)*

[1] Middle English (in the sense ‘make, shape, appearance' also ‘a particular make or style’”: from Old French façon, from Latin factio (n-), from facere ‘do, make’ (Oxford Dictionaries, s.f.).

De acuerdo con Wikipedia (2018), moda es un conjunto de prendas de vestir, adornos y complementos que se basan en gustos, usos y costumbres, y que se utilizan por una mayoría durante un periodo de tiempo determinado y que marcarán tendencia. Otra definición es la hallada en el *Garzanti della lingua italiana*, que define la moda como: El uso más o menos cambiante que, al convertirse en el gusto dominante, se impone en las costumbres, en las formas de vida, en las maneras de vestir. Resulta inevitable citar la definición académica de Margarita Rivière que define la moda como "lo actual, lo que está en vigor e interesa a una mayoría en un momento determinado. Aplicada a la indumentaria, moda es aquel atuendo, estilo, prenda, color o complemento que se lleva por parte del grupo socialmente más importante o hegemónico, que es capaz de influir en los demás" (Rivière, 1996, 196).

De estas definiciones se destaca la característica tiempo dentro de la moda, la moda es efímera, algo que dura un determinado periodo de tiempo y se trata de un fenómeno seguido por la mayoría. Sin embargo, sería demasiado simplificador aproximar la moda desde dicho ángulo ya que atendiendo a otros criterios, la moda también hace referencia a otras necesidades más básicas como la funcionalidad para protegerse y tener seguridad. El ser humano se viste para protegerse de las agresiones del entorno y resguardarse del clima (Deslandres, 1987). Además hay otra necesidad humana en la moda, una necesidad social y psicológica (Gabarrón, 1989) en la que la estética predomina sobre el beneficio funcional. También, de acuerdo con Corbellini y Saviolo (2012), la moda consiste en un sistema de comunicación no verbal que muestra la ocupación, el rango, género, origen, salud y el grupo de afiliación.

A modo de conclusión se puede decir que la moda no se vincula solo con el vestir, es un fenómeno efímero, tiene necesidades funcionales pero a la vez cubre una necesidad social que ayuda a transmitir la identidad de la persona.

2.2. Origen del fenómeno moda

La moda, como tal, se puede decir que surgió antes de mediados del siglo XIV, periodo histórico en el que por primera vez se impone un modelo de vestimenta exclusivo para el hombre y otro para la mujer. Un hito en la historia, porque aunque hoy en día se encuentre algo normal que chicos y chicas vistan de diferente modo como si siempre hubiese estado establecido así, durante muchos siglos la ropa fue común al sexo masculino y femenino. Por ejemplo, en el antiguo Egipto hombres y mujeres llevaban un vestido-túnica idéntico y para esta civilización el concepto de moda para ellos era inexistente (Squicciarino, 2012).

A partir de mediados del siglo XV se abandona la túnica para empezar a usar dos piezas, el jubón (una especie de chaqueta corta) y unos calzones ceñidos a las piernas. La mujer, en cambio, experimentó un cambio menor, en cuanto a prendas, puesto que continuó vistiendo de largo, pero sus ropajes se volvieron más atrevidos, comenzaron a diseñarse con el objetivo de favorecer el cuerpo de la mujer. En el siglo XVII las clases más altas de Francia para diferenciarse de la corte española comenzaron a vestirse según el gusto francés (*a la mode*) y desde entonces la

palabra se ha ido usando para designar los continuos cambios que verifican el campo del vestido (Squicciarino, 2012).

La moda actual, tal y como la conocemos en nuestros días, es un fenómeno reciente. Después de la segunda guerra mundial y hasta los años 70, el concepto de moda estacional estaba sólo relacionado con la ropa, y en particular, con un segmento concreto de ropa femenina: la *haute couture y el ready-to-wear*. En los años 90 es cuando se puede hablar de la industria de la moda como un fenómeno global.

Como conclusión, cabe destacar que el poder de la industria de la moda concebida como se ha definido previamente, es de tan gran intensidad que desde los años 80, esta concepción de la moda se ha extendido a otros segmentos como las gafas, los perfumes, los accesorios (relojes, joyas, bolígrafos, etcétera), los móviles, los destinos turísticos… provocando una renovación de los mismos cada temporada. No existe, hasta la fecha, otro campo de la actividad humana donde el cambio sistemático de producto se haya institucionalizado tanto como en la ropa y en los bienes de consumo cercanos a ésta (Corbellini y Saviolo, 2012).

2.3. Clasificación de la industria de la moda

Sorprende el gran número de fabricantes y marcas de moda que existen en el mercado para los consumidores. Cada marca atiende un segmento de mercado. De hecho, en la literatura se pueden encontrar varias clasificaciones de la industria de la moda. Una de las más aceptadas es la de los autores Saviolo y Testa (2005) que distinguen cinco segmentos en función de dos variables: precio y calidad de los productos. Estos segmentos son: a. *Haute Couture* (o *High Fashion*, Moda de alta costura); b. *Prêt-à-porter (Ready to Wear* o listo para llevar); c. Difusión; d. *Bridge o* Novias y; e. *Mass Moving* (Moda rápida). En general, esta clasificación se simplifica en tres categorías: alta costura, *prêt-à-porter* y fabricación en serie o moda para masas (Dillon, 2012)

La **alta costura** se basa en el diseño y máxima calidad. Las colecciones de las casas de moda se fabrican en exclusiva para clientas privadas (Brun y Castelli, 2013). La calidad y la imagen de marca son centrales (Bandinelli et al., 2013) pero también es crítico para el éxito es la innovación, es decir, la idea creativa, la invención artística del diseñador (Brun y Castelli, 2013). La producción es muy reducida, el tiempo y dedicación precisan una mano de obra altamente cualificada. La producción oscila entre diez y veinte piezas por modelo lo cual implica un elevado coste. Estas colecciones se exhiben dos veces al año con unos 35 modelos de noche y día (Cabrera y Frederich, 2010). Para que una marca sea considera de alta costura, su casa de diseño ha de ser invitada a incorporarse a la *Chambre Syndicale de la Haute Couture*. Es este un organismo regido por el Ministerio de Cultura francés, ubicado en París y compuesto por un comité de diseñadores. Actualmente, esta cámara está compuesta por unas veinte casas, entre ellas Armani, Christian Dior o Chanel, que son los primeros interesados en hacer cumplir la estricta normativa de la organización. Un miembro de esta cámara de Alta Costura debe: a) Diseñar ropa a medida para clientes particulares, con una o más pruebas; b) Mantener un atelier (estudio de diseño) en

París que emplee al menos a quince personas, c) Presentar ante la prensa francesa, dos veces al año, una colección de al menos treinta y cinco modelos.

El ***Prêt-a-porter*** o ***Ready to Wear*** (listo para llevar) surge en 1960. Se considera que fue Yves Saint Laurent (YSL), en 1966, el primer modisto de alta costura que presentó una línea completa de prêt-à-porter, abriendo sus tiendas en la Rive Gauche, de París. La intención implicaba un deseo de democratizar la moda (Grose, 2012). El *prêt-à-porter* se refiere a prendas manufacturadas en contraposición a manuales. Las telas y acabados son de alta calidad. Algunas colecciones son limitadas y exclusivas por ello el precio es alto pero por debajo de la alta costura. Se consideran las *trensetters*, es decir, las marcas que marcan la tendencia. Al igual que la alta costura se exhiben dos veces al año en las semanas de la moda que se realizan 6 meses antes del envío a las tiendas.

La **fabricación en serie** o ***Fast Fashion*** *(*moda rápida) representa la línea de producción más industrializada y, por lo tanto, con un precio inferior. Las técnicas de fabricación en serie se inventaron a finales del XIX pero no es hasta después de la Segunda Guerra mundial cuando se imponen sobre la alta costura. Los diseños de este segmento se basan en tendencias generalizadas y buscan su inspiración en las colecciones de *prêt-à-porter*, con el objetivo de tener una venta ágil de las prendas. A este segmento también se le conoce con el nombre de “moda pronta”. El objetivo de cadenas como Zara, Mango o Primark es producir ropa de tendencia en muy poco tiempo, logrando reponer colecciones en no más de seis semanas. Realmente la moda pronta ha sido la gran revolución de las cadenas de moda.

Otra clasificación que merece la pena referencia es la del autor Sen (2008) que distingue entre: a. Producto básico; b. Producto Estacional y; c. Producto de moda. Los productos básicos se venden a lo largo de todo el año y representan aproximadamente el 20% del mercado. Los productos estacionales suelen tener un ciclo de vida de 20 semanas y aproximadamente representan el 45% del mercado. Por otro lado, los productos de moda, tienen un ciclo de vida de 10 semanas y representan alrededor del 35% del mercado.

2.4. Tendencias de la industria de la moda

Se puede afirmar que desde los años 90 la moda se convierte en un fenómeno global y por ello el sector de la moda representa en la actualidad uno de los sectores más dinámicos y crecientes. El sector moda como tal comprende distintas actividades que van desde la transformación de materias primas naturales como el algodón o artificiales como el poliéster hasta la producción de hilos y telas. Además abarca distintas actividades productivas como la confección de ropa, calzado y accesorios, joyería, relojes y cosmética. Según la consultora Euromonitor (2018) en el año 2017, las ventas de ropa y calzado crecieron un 4% a nivel global respecto al año anterior, representando 1,7 billones de dólares (1,4 billones de euros) y se pronostica que siga creciendo un 2% en el año 2022. Por segmentos, la ropa y el calzado deportivo fueron las que más crecieron a un ritmo por encima del 6% (véase Anexo 1). A ello le siguió

el sector infantil. Además, es oportuno señalar que el 16% de la facturación de ambas categorías tuvo lugar en el comercio electrónico frente al 10% del ejercicio anterior.

En Europa el gasto en prendas de vestir crece a lo largo de los años y la UE28 gasta unos 511 billones de euros en el año 2017 frente a 503 según Statista (2018). A nivel gasto en prendas de vestir cinco países lo lideran: China, Estados Unidos, Reino Unido, Alemania y Japón (Anexo 2). De este dato es destacable como China de ser la fábrica textil del mundo ha pasado a ser también el primer consumidor del mundo poniendo de manifiesto la evolución de esta nación que es patente en otros sectores como el de telecomunicaciones con el gigante chino Huawei o el comercio electrónico de la mano de AliExpress. En cuanto al crecimiento en el sector textil sobresalen las naciones asiáticas como Irán, Indonesia, Vietnam, Corea del Sur, India, Uzbekistan que se caracterizan por su gran población y riqueza de materias primas.

Aunque Francia e Italia resulten los referentes mundiales en moda, en alta costura y *prêt-a-porter* respectivamente, España se ha convertido en un país de referencia en el mundo gracias al *fast fashion* cuyos máximos exponentes son Inditex y Mango. Ello explica que la moda española sea un sector creciente y dinámico. En España hay 6.800 empresas dedicadas al sector y eso explica su alta fragmentación. La moda en España, en la que se incluye confección, complementos, calzado y joyería, en el año 2016 y 2017 representó un 2,9% del PIB lo cual supone unos 30.000 millones de euros. Esta cifra convierte al sector de la moda como uno de los pilares económicos más importantes de España siendo mayor que todo el valor añadido bruto. Dado que el sector de la moda comprende distintas actividades, este dato si se desagrega muestra que la parte comercial, es decir distribución comercial y minorista, es la que mayor impacto económico tiene en España. Un 13.8% del PIB procede del comercio de la moda frente al 5.6% de la parte manufacturera en el sector secundario. El sector de la moda en España genera el 4.1% de los empleos del país. El desglose de esta cifra muestra que la parte comercial es tres veces mayor que la parte industrial indicando que España es más potente en el *retail* o distribución frente a la parte de producción. En términos de exportaciones ocupa la posición cuarta con un 8.4% del total siendo esta cifra creciente. Francia e Italia lideran la recepción. En cuanto a importaciones, un 8.9% proveniente en su mayoría de materia prima asiática.

A pesar de que la crisis del año 2007 causó mella en el sector, la moda española ha ido recuperándose poco a poco y los últimos datos del mes de mayo del año 2019 indican un crecimiento del 2.9% quedando el acumulado anual con un incremento de las ventas del 2.3% (Acotex, 2018). Este dato debe ser tomado con cautela porque las ventas del mes de abril cayeron un 8.4% y el mes de mayo de 2017 también fue decreciente además la climatología de junio no está animando al consumo, todo ello junto con los datos macroeconómicos pueden debilitar el sector en el conjunto del año 2019 (Véase en Anexos 3, 4, 5 y 6 una breve reseña a datos sobre la industria). En España los distribuidores principales son Inditex (22 millones de facturación) y Mango (2 millones de facturación).

Para concluir, el último informe de la consultora McKinsey resumen la evolución en el año 2019 del sector de la moda destaca algunos hechos significativos para tener en cuenta:

- Situación de la economía global: un 70% de los directivos encuestados piensan que hay que ser cautelosos con la situación global y prudentes ante una posible recesión en el año 2020. Recomiendan vigilar la productividad ante todo. Una nuevo mercado masivo aparece en el escenario: India con más de 690 millones de usuarios de *smartphones* en el año 2022 lo que supone 2.3 veces más que en el año 2017.
- Cambios en el comportamiento del consumidor: los consumidores quieren acceder a la moda, a cualquier tipo, ello explica la emergencia de nuevos modelos como los de alquiler de ropa, segunda mano, reparación etc. La propiedad de la prenda puede desaparecer. Por otro lado, hay una mayor preocupación por la moda sostenible y se demandará más a los fabricantes en materia de responsabilidad social. El comercio electrónico y gigantes como Amazon conducen a una filosofía de compra de ropa de "ahora o nunca" y la disponibilidad en la web será decisiva.
- Modelo de negocio: se requiere mayor innovación en toda la cadena de valor para lograr satisfacer a tiempo la demanda y reducir los stocks. Además esta innovación debe de ser ética.

2.5. La naturaleza de la moda y su gestión empresarial

Es evidente que la moda se refiere a vida corta y es la característica más diferenciadora de su naturaleza pero hay otros elementos que describen la naturaleza de la moda (Christopher et al., 2004) como se muestra seguidamente: 1. Ciclos de vida cortos: el producto suele ser efímero, diseñado para capturar el momento, el periodo de venta es muy corto y cíclico, se mide en meses o incluso semanas; 2. Alta volatilidad: la demanda de estos productos rara vez es estable o lineal. Puede estar influida por el clima, películas, famosos, futbolistas; 3. Baja capacidad de predicción: dada la volatilidad de la demanda, es muy difícil predecir con seguridad la demanda total en un periodo, y aún más por semana y por artículo y; 4. Compra impulsiva: muchas decisiones de los consumidores de estos productos se hacen en el punto de venta, por ello es crítica la disponibilidad del producto. Estas características junto al carácter marcadamente competitivo de la industria de la moda explican que la gestión del negocio sea de gran complejidad. La planificación es larga, compleja e inflexible (Jones, 2002; Hines, 2004). A ello hay que sumarle el cambio en los consumidores que cada vez demandan un surtido muy variado. Es por ello que los fabricantes se ven en la necesidad de variar el catálogo de productos constantemente y extender el número de estaciones encontrándose casos extremos como el de Zara que puede presentar hasta treinta estaciones en un año. Las implicaciones son cuantiosas para todos los departamentos, desde marketing hasta finanzas por lo que la gestión de la cadena de valor se hace extremadamente complicada. El hecho de ser un negocio poco predecible hace que logísticamente sea un reto. Es necesario hacer predicciones muy afinadas para evitar el sobrestock o la falta de stock por el contrario que sería igual de nefasta. La variable tiempo en el modelo de negocio es crítica. Ello ha dado lugar a movimientos para mejorar la respuesta de las cadenas de valor en la industria de la moda con la introducción de conceptos como el *just-in-time* (Bruce et al., 2004),

la cadena de valor ágil (Christopher et al., 2004; Bruce et al., 2004) y los sistemas de rápida respuesta (Giunipero et al., 2001; Fernie and Azuma, 2004). Estos se basan en la filosofía del *Quick Response* (QR) surgida en 1984 de un programa textil en Estados Unidos. Este movimiento ha sido la raíz de la denominada *Agile Supply Chain* de gran interés (Christopher, 2000). Es principal objetivo de la QR es reducir todos los lapsos que ocurren en la cadena de valor.

Como eslabón de la cadena de valor, la función de marketing es determinante para poder crear una oferta atractiva el mercado. El marketing de la moda se caracteriza por su alto nivel de sofisticación así como un uso particular de las herramientas de marketing mix. Son varios los elementos de marketing mix y a modo general se refieren a producto, precio, distribución y promoción (conocidos como las "Ps").

El producto es la prenda que el fabricante comercializa. En una prenda de ropa el producto tiene componentes tangibles e intangibles así como beneficios funcionales y emocionales. Son componentes tangibles el diseño, elemento determinante en el mercado de la moda, el tipo de material con el que se fabrica la prenda, el corte o la terminación. Además de un claro beneficio funcional como por ejemplo abrigar, los productos en moda ofrecen un beneficio emocional único que cubre necesidades y deseos como de autoafirmación. Gracias a las marcas, los productos son registrados y dotados de un valor que se convierte en uno de los activos intangibles más importantes de una empresa. Esta marca puede corresponder al fabricante o no coincidir con la misma denominación. Un ejemplo es Inditex y Zara.

La distribución permite que el producto se entregue al consumidor final. Es un elemento que se ha revolucionado gracias a la llegada del comercio electrónico, canales cada vez más poderosos para comercializar la ropa. Sin embargo, la tienda física sigue siendo indispensable porque contribuye a la creación de la imagen de marca y a ofrecer un servicio único así como una experiencia de compra.

La comunicación y promoción es determinante en el universo de la moda. Es una de las industrias donde más se invierte en publicidad y relaciones públicas para construir la imagen de marca y promocionar las prendas. En los últimos años, *instagram*, blogs y otros canales digitales se han convertido en los mejores escaparates de la moda.

En un mercado como la moda tan volátil, tan corto de tiempo, es indispensable la gestión del precio y sus descuentos para garantizar la venta del producto y poder liquidar los stocks al cierre de la temporada. Es por ello que el precio y el tema de interés de este trabajo, el descuento en precio, resulten una de las decisiones estratégicas más relevantes en esta industria. Todo precio se fija de acuerdo a dos niveles: el precio más alto que está dispuesto el consumidor a pagar por el producto y el precio más bajo posible para la empresa para poder cubrir costes. Una de las características del precio es que es la herramienta más flexible de una decisión de marketing y puede actuarse sobre la misma de modo inmediato. No ocurre así cuando ya se ha lanzado el producto, se ha distribuido y se ha comenzado la campaña de comunicación. Además el precio es el elemento de marketing que tiene un impacto directo sobre los ingresos de la empresa.

CAPITULO III. BIG DATA Y LA INDUSTRIA DE LA MODA

Este capítulo tercero se centra en el fenómeno del Big Data. Se ha distribuido en dos epígrafes. El primero es una breve introducción al origen y al concepto del Big Data y concluye con unos datos sobre las tendencias del Big Data en el mundo. El segundo apartado muestra los trabajos hallados en la literatura académica sobre los fenómenos Big Data y moda que como se puede apreciar son escasos a pesar de la emergencia del tema.

3.1. Origen, definición y conceptualización de Big Data

El fenómeno Big Data es uno de los más predominantes en la actualidad y es ubicuo. En todas las partes y lugares se habla de ello. Representa a la vez una de las mayores oportunidades y al mismo tiempo retos. Cada día el mundo produce alrededor de 25 pentallones[2] de bytes de datos (Dobre y Xhafa, 2014) siendo el 90% de estos datos no estructurados. En el año 2020 se generarán y consumirán más de 40 zettabytes, es decir, 40 trillones de gigabytes de datos (Gantz y Reinsel, 2012). Toda esta desmesurada cantidad de datos de naturaleza compleja, heterogénea y proveniente de todas partes, cualquier dispositivo, sensor y tiempo explica la era actual de los datos. Independientemente de cómo son los datos y la fuente, el mayor reto del BD es cómo analizar los datos para que sean transformados en gran valor (*Big Value)*. Es precisamente este valor potencial del BD lo que da lugar a que el BD se conozca como el petróleo digital (Yi et al., 2014) o la nueva materia prima del siglo XXI (Berners-Lee y Shadbolt, 2011). El desafío para transformar la masiva cantidad de datos es que el entorno computacional no evoluciona a la misma velocidad que se generan los datos y las tecnologías existentes de almacenamiento y gestión de datos no son adecuadas. Nuevos desarrollos tecnológicos en bases de datos como MapReduce, Hadoop o NoSQL Database están permitiendo el procesado de estos datos para extraer conocimiento que son aplicados a diversos sectores desde el sanitario, energético, económico y climatológico (Yi et al., 2014).

Las investigaciones sobre BD son muy abundantes y se encuentran en constante crecimiento (Sivarajah et al., 2017). Gran parte de los estudios son de naturaleza analítica mostrando algoritmos, simulaciones o modelos matemáticos para procesar el BD. Todos estos trabajos coinciden es que el BD cuando se gestiona, procesa y analiza de forma adecuada, genera conocimiento novedoso que facilita la toma de decisiones (Jukic et al., 2015). Una parte de los estudios se dedica a estudiar las oportunidades del BD no solo en el mundo empresarial pero también científico (p.ej. Chen y Zhang, 2014) y por otro lado, un buen número de trabajos abordan los retos del BD derivados de la complejidad de los datos (Gandomi y Haider, 2015), de la falta de talento especializado en BD (Kim et al., 2014), problemas de seguridad y privacidad (Barnaghi et al., 2013) o el problema de utilizar el análisis estadístico tradicional en el BD (Sandhu y Sood, 2014; Zhang et al., 2015).

[2] 1 exabyte equivale a 1 pentabyte o 1 billón de gigabytes

Como concepto el BD es relativamente reciente, sus orígenes son inciertos y aún sigue existiendo un debate en la comunidad sobre la definición del mismo dada la variedad de definiciones. Según Diebold (2012) el fenómeno apareció a mediados de la década de los 90 en una comida de empresa Sillicon Graphics siendo John Mashey el primero en acuñar el término pero no es hasta el año 2011 cuando se empieza a extender debido fundamentalmente por el apoyo de IBM y otras empresas líderes en tecnología (Gandomi y Haider, 2015). El hecho de que las divulgaciones sobre Big Data hayan sido lideradas principalmente por comunidades de tecnología de la información, las aproximaciones al término se han centrado fundamentalmente en tecnología y lo cierto es que el Big Data no es una tecnología (TechAmerica Foundation, 2018).

Como indica la denominación *Big*, una definición muy simple podría ser "gran cantidad de datos" y por ello el volumen es la característica más sobresaliente del Big Data y la primera en mención. No cabe duda que el volumen de datos y las fuentes de los mismos crecen exponencialmente y se estima que se generan 2.5 exabytes (1 exabyte=1 millón de terabytes) al día de datos (IBM, 2015). A medida que ha ido evolucionando el concepto se han ido esclareciendo otras dimensiones características del Big Data siendo el Modelo de las Tres Vs (Volumen, Variedad, Velocidad) de Laney (2001), el marco común para definirlo (Chen et al., 2012; Kwon et al., 2014) como muestran las definiciones de la consultora Gartner (2018) y de la TechAmerica Foundation. Este marco conceptual define Big Data como los datos que presentan las siguientes características como muestra la figura posterior y que explican los retos que conlleva su gestión, procesamiento y análisis:

-*Volumen:* se refiere a la cantidad masiva de los datos, pentabytes, exabytes y zetabytes, cantidades que no pueden ser procesadas por un humano pero sí por una máquina. Un estudio de IBM en el año 2012 cifraba que a partir de un terabyte se puede considerar Big Data. Eso equivale a almacenar 1500 CDs o 16 millones de fotos de Facebook. El tamaño de los datos varía dependiendo de la industria y se requieren distintas tecnologías de gestión de datos ya que no es lo mismo por ejemplo una base tabulada versus datos de video. La principal causa de esta cantidad de datos masiva es el crecimiento exponencial de las fuentes de datos y sensores de alta resolución. Son necesarias nuevas técnicas de minería de datos y aproximaciones ante tal magnitud (Zhao et al., 2013).

-*Variedad*: los datos provienen de múltiples y diversas fuentes por todas partes: desde sensores de tráfico, de clima, información de pasajeros, comentarios en medios sociales como Facebook y Twitter, fotos digitales como Instragram y vídeos como Youtube, registros de transacciones bancarias, señales de móviles, direcciones IP de wifi y un largo etcétera. Los datos no vienen en una forma homogénea, solo el 5% de la información actual es estructurada o tabulada en hojas de cálculo o bases relacionales. La información no estructurada como email, video, blogs, conversaciones de un *call center* o de los medios sociales, suman aproximadamente el 85% de los datos que se generan en la actualidad. Por ello, se hace imprescindible combinar datos estructurados y no estructurados que permitan el procesamiento que suponen un gran reto (Labrinidis y Jagadish, 2012).

-*Velocidad*: se refiere a la tasa en la cual los datos son generados y la velocidad a la que deben de ser procesados, en tiempo real. Ello se debe a la proliferación de dispositivos móviles y sensores. Ello representa el reto no solo de crear los datos sino también actualizarlos a la vez (Chen et al., 2013). Un ejemplo es el distribuidor estadounidense Walmart que genera más de un millón de transacciones cada hora (Cukier, 2010). Esta información proporciona información sobre los clientes, su situación geoespacial, sus preferencias, patrones de consumo que son de gran valor para diseñar ofertas personalizadas (Gandomi y Haider, 2005).

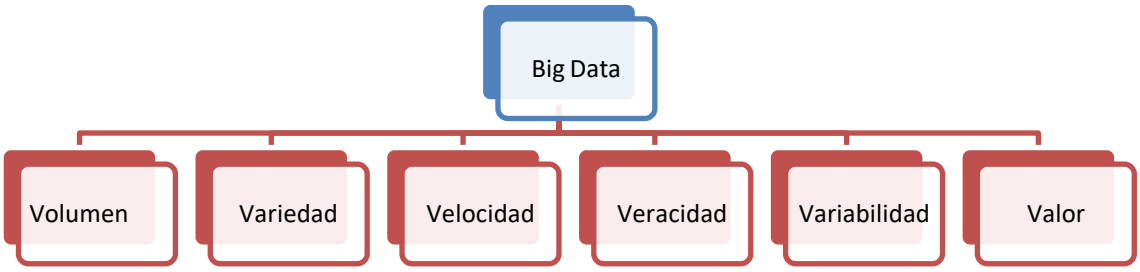
Además de este modelo conceptual de las 3Vs, se han ido añadiendo otras dimensiones (Gandomi y Haider, 2015): 1. Veracidad; 2. Variabilidad y; 3. Valor.

-*Veracidad:* fue acuñada por IBM como la cuarta V y representa la falta de fiabilidad e inconsistencia de algunos datos. Por ejemplo, los análisis de sentimiento en las redes sociales que son muy valiosos para conocer a los clientes pero inciertos por su naturaleza subjetiva. Ello requiere que existan nuevas herramientas para combatir la imprecisión de los datos. No se trata solo de calidad de los datos pero también de comprenderlos (Gandomi y Haider, 2015).

-*Variabilidad* y complejidad: fueron introducidas por la consultora SAS como dos dimensiones adicionales del Big Data. La variabilidad, que se confunde con la variedad de los datos, indica la tasa de variación del flujo de datos. A menudo, la velocidad de los datos no es consistente y tiene picos. Se refiere a que los datos ofrecen distinto significado. Por ejemplo, un análisis de sentimiento en de un tweet pueden ser interpretados de distinta manera según el contexto (Zhag et al., 2015).

-*Valor:* fue introducido por Oracle y lo define como “baja densidad en valor”. Los datos recibidos en su forma original presentan bajo valor relativo en comparación a su volumen. Almacenar el BD es complejo y vistos de forma independiente los datos pueden parecer insignificantes (Zaslavsky et al., 2012) además extraer el valor puede resultar muy costoso (Abawajy, 2015).

Ilustración 1. Las dimensiones del Big Data



Fuente: elaboración propia a partir de Gandomi y Haider (2015)

3.1.1. Big Data tendencias

Se puede decir que el Big Data y su utilización como herramienta para optimizar los negocios se ha popularizado en el año 2018. Es estimado que el mercado de Big Data y *business analytics* generará unos ingresos de 260.000 millones de dólares en el año 2022, un incremento interanual del 11,7 % desde el año 2017 según la consultora IDC (2019). Está claro que a esta tecnología aún le queda mucho recorrido. En 2019 según la empresa *Analytics Insights* cobrarán relevancia los siguientes fenómenos:

- Internet de las Cosas (IoT): El crecimiento de objetos conectados obligará a las empresas a dotarse de nuevas infraestructuras.
- Análisis predictivo: Permite a las empresas identificar riesgos y oportunidades futuras.
- Computación cuántica: la potencia de procesamiento de las computadoras tradicionales es limitada. Las máquinas cuánticas procesarán los algoritmos mucho más rápido. Cuando sea una realidad subirá el nivel el Big Data a cuotas inimaginables. En la carrera por lograr el primer ordenador cuántico se encuentran Intel, Google, IBM y Microsoft, entre otras.
- Programas de código abierto: El software libre permite reducir costes y, por tanto, que más empresas se beneficien del Big Data. Hadoop es una de las herramientas más conocidas, junto con Elastic Search, MongoDB y Cassandra.
- Análisis de “datos oscuros” (*dark data*): Se denomina así a los datos que las empresas no pueden analizar, ya que, por distintas razones, no les es posible capturarlos. Aparecerán nuevas soluciones para su captura.
- *Edgecomputing*: Esta tecnología permite saber qué datos es mejor almacenarlos localmente y cuáles en la nube. Esto acelera el proceso de análisis y, por tanto, la toma de decisiones.
- Detección de amenazas: Con el auge de IoT, el análisis de datos prolifera como una herramienta para predecir y detectar las amenazas de ciberseguridad, integrándose en la estrategia para luchar contra el robo de datos.
- Importancia del *Chief Data Officer* (CDO): El máximo responsable de los datos en una empresa desempeña un importante papel: maximizar el valor de los datos. El 90 % de las grandes compañías tendrán esta figura en 2019, según Gartner.

3.2. Big Data y la industria de la moda

Con el propósito de comprender y proporcionar conocimiento sobre el fenómeno del BD en la industria de la moda, se realizó una búsqueda de los trabajos existentes sobre el tema de interés. Para identificar los artículos relevantes se utilizó la base de datos Scopus. La razón para elegir esta base de datos es que tiene una cobertura de casi más de 18.000 títulos de entre más de 500 publicaciones internacionales incluyendo alrededor de 16.500 revistas con revisión a pares de distintas áreas. Por consiguiente permite buscar y localizar un número significativo de

publicaciones sobre cualquier temática. Además a la búsqueda se le impusieron una serie de restricciones para garantizar la calidad: a. Solo se contemplaron artículos con revisión por pares; b. Artículos en inglés y español aunque de este idioma no se identificó ninguno; c. Artículos desde el año 1996 hasta marzo de 2019, año 1996 porque es cuando surgen las primeras publicaciones de Big Data; d. Se seleccionaron las áreas *Business and Management, Computer Sciences, Decision Sciences y Social Sciences*; e. Se excluyeron documentos como actas de conferencia, libros, capítulos de libros, editoriales, publicaciones comerciales. Solo se incluyeron artículos científicos y artículos en prensa; f. Se contemplaron tanto artículos empíricos como conceptuales y; g. Para la selección se utilizaron buscadores lógicos[3] que contuvieran palabras claves en el título, resumen, palabras claves y en todo el texto posteriormente.

La búsqueda arrojó solo 20 artículos que reúnen el fenómeno Big Data y la industria de la moda, 10 de esos artículos se han publicado en los años 2018 y 2019 siendo un tema de investigación en emergencia. Fundamentalmente se observa que estos trabajos se centran en tres áreas:

- ✓ Predicción de demanda en particular a través de redes sociales y comentarios (p.ej. Hu, M. et al., 2019; Silva, E.S., et al., 2019; Ngai, E.W.T. et al., (2018). Estos trabajos utilizan principalmente los comentarios en redes sociales para la predicción de ventas y de las futuras tendencias. También aparecen artículos sobre predicción de ventas a través de Google Trend que ya se ha aplicado a industrias con la del automóvil y la vivienda.

- ✓ Optimización de la cadena de valor (p.ej. Majeed et al., 2017; Choi, 2007). Debido a que el caso Inditex es uno de los más analizados en el área de operaciones porque la cadena de valor de esta compañía es ejemplo de eficiencia, varios estudios utilizan el Big Data para realizar control de stock de producto, aprovisionamiento para agilizarla. Otro campo de aplicación de Big Data es alcanzar una mayor sostenibilidad en la cadena de valor mediante el estudio de los materiales que componen las prendas.

- ✓ c. Clasificación y predicción con imágenes (p.ej. Guan C. et al., 2016; McAuley J. et al., 2015). Una rama de las investigaciones trata de usar el análisis de imágenes, en especial, en instagram para predecir la tendencia de la moda a través de los atributos de la prenda.

Finalmente se detecta el uso de la inteligencia artificial como rama de ingeniería aplicada al punto de venta. Un ejemplo es el desarrollo de probadores inteligentes o de tecnologías a través de imágenes para probarse la prenda de ropa.

[3] Como palabras claves se utilizó: Big Data AND Fashion OR Big Data AND Textile OR Big Data AND Clothing OR Big Data AND Garment OR Big Data AND Fashion Retailer

CAPITULO IV. APRENDIZAJE AUTOMATICO

Este capítulo expone los conceptos principales detrás del aprendizaje automático (*Machine Learning*), de ahora en adelante ML. Gracias a la cantidad de datos tanto estructurados como no estructurados así como el desarrollo computacional y tecnológico, el ML es objeto desde hace 10 años de desarrollo exponencial en todas las industrias y se encuadra dentro de la disciplina de inteligencia artificial permitiendo aprender a partir del reconocimiento de voz, reconocimiento de imágenes y ofreciendo nuevas soluciones como el coche autónomo. Este capítulo se organiza en seis apartados. El primero introduce el concepto de aprendizaje automático, el segundo su clasificación. A continuación se ahonda en el aprendizaje supervisado para describir en el apartado cuarto la biblioteca Scikit-Learn de Python. El apartado quinto hace referencia a las métricas para evaluar un algoritmo de clasificación. Seguidamente se describen los tipos de algoritmos de aprendizaje automático.

4.1. Machine Learning o Aprendizaje automático

En el año 2006, Hinton et al., publicaron un trabajo mostrando cómo entrenar una red neuronal capaz de reconocer dígitos de escritura manual con una precisión perfecta, mayor del 98%. Lo denominaron *Deep Learning*. Entrenar una red neuronal en ese momento se consideraba imposible y la mayoría de los investigadores habían abandonado la idea en la década de los 90. Este trabajo propició el interés por el *deep learning* y otras áreas del aprendizaje automático o *machine learning* (ML) a cuyo desarrollo ha contribuido las grandes cantidades de datos de que se disponen así como la potencia computacional. Desde entonces la industria ha sucumbido al ML y está presente en casi todas desde reconocimiento de voz, rankings de páginas webs hasta pronto ser capaz de que los coches conduzcan solos.

La mayoría de las personas asocian el ML a un robot como si fuese algo del futuro. Sin embargo, este fenómeno ya lleva décadas instalado y presente en diversos campos muy especializados como el *Optical Character Recognition* (OCR). Se puede decir que la primera aplicación de ML que se generalizó fue en los años 90 con el *spam filter.*

Según O'Reilly, ML es la ciencia y el arte de programar ordenadores para que puedan aprender de los datos. Una definición más técnica es la de Tom Mitchell en 1997: se dice que un programa de ordenador aprende de la experiencia E respecto a una tarea T y algunas medidas de resultado p, si su resultado en T, medido como P, mejora con la experiencia E. Por ejemplo, el filtro *spam* aprende de los datos. La tarea T es etiquetar los *spam* de entre los nuevos emails, E es la experiencia, el entrenamiento y P medida del rendimiento es el porcentaje de emails bien clasificados.

La ventaja de usar ML es que los problemas son complejos y los programas tendrían que tener una larga de reglas complejas difíciles de ejecutar. Gracias al ML se simplifica el proceso, más sencillo de ejecutar y más preciso frente a la aproximación tradicional. Por consiguiente ML es adecuado para:

a. Problemas cuya solución requiera gran trabajo manual o una larga lista de reglas, gracias al ML un algoritmo lo simplifica (Tang et al., 2014).
b. Problemas complejos que no tienen solución con un método tradicional.
c. Entornos fluctuantes que tienen datos continuos y variables (Domingos, 2012)
d. Problemas que surgen cuando se tienen tantos datos que no se sabe obtener valor.

4.2. Tipos de Machine Learning

Hay multitud de sistemas de ML que se clasifican en varias categorías: 1. Si son o no entrenados con la supervisión humana (supervisado, no supervisado, semisupervisado y aprendizaje por refuerzo); 2. Si pueden o no aprender incrementalmente sobre la marcha (*online* vs. *batch learning*) y; 3. Si pueden trabajar simplemente comparando puntos de datos nuevos con puntos de datos conocidos, o detectar patrones en los datos de entrenamiento y construir un modelo predictivo (*instance-based* vs. *model-based learning*). Todos estos tipos de ML pueden ser combinados entre sí. Con el ánimo de no extenderse más allá de lo estrictamente necesario para este trabajo se muestra en más detalle la diferencia entre aprendizaje supervisado y no supervisado.

En el *aprendizaje supervisado o con clase* los datos de entrenamiento para el algoritmo incluyen soluciones deseadas que se denominan etiquetas. Dado que se sabe cuál es la salida deseada, por ello se utiliza el concepto de "supervisado". Un problema típico de este tipo de aprendizaje es el de clasificación. Un ejemplo es el filtro de mensajes a spam, el algoritmo se entrena con muchos emails cada uno en una clase si es spam o no y la máquina ha de aprender a clasificar nuevos emails. Otro trabajo similar es predecir una variable objetivo de tipo numérico como el precio de una casa dado un conjunto de atributos como la edad o el barrio que se denominan predictores siendo este tipo de tarea una regresión. Para entrenar el sistema se requiere dar a la máquina muchos ejemplos de casas incluyendo sus predictores y etiquetas (precio). Algunos de los algoritmos de aprendizaje supervisado más importantes son: regresión lineal, *k-Nearest Neighbors*, regresión logística, *Support Vector Machines* (SVMs), árboles de decisión, *random forest* y redes neuronales.

En el *aprendizaje no supervisado o sin clase*, los datos de entrenamiento no están etiquetados o con estructura, se asume que todas las observaciones son causadas por variables latentes siendo la finalidad describir la estructura de los datos organizándolos. No se dispone de información previa que ayude a la identificación de patrones o determinar a qué clase pertenecen los datos. El sistema trata de aprender sin nadie que lo dirija. Este aprendizaje persigue crear distintos subconjuntos basándose en la similitud de las características del conjunto de datos de partida. Entre los algoritmos más importantes de este tipo de aprendizaje se encuentran: el *Clustering* (*K-Means, DBSCAN, Hierarchical Cluster Analysis* (HCA)); *Anomaly detection and novelty detection ((One-class SVM, Isolation Forest*); reducción de dimensionalidad y visualización (*Principal Component Analysis* (PCA), *Kernel* PCA,

Locally-Linear Embedding (LLE), *t-distributed Stochastic Neighbor Embedding* (t-SNE) y; *Association rule learning* (A priori, Eclat).

La siguiente figura muestra las diferencias entre los dos tipos de ML:

Ilustración 2. Aprendizaje supervisado y no supervisado

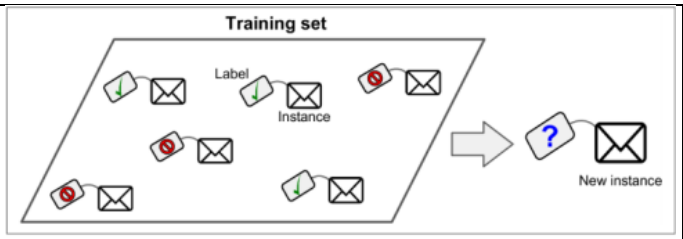


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

Figure 1-7. An unlabeled training set for unsupervised learning

Fuente: O'Really (2018)

4.3. Aprendizaje supervisado y Clasificación.

El aprendizaje automático supervisado, objeto del presente estudio, a su vez tiene dos variantes: clasificación y regresión (Bishop, 2007). Estas dos aproximaciones de aprendizaje dependen del objetivo de aprendizaje. Si el resultado deseado consiste en una o más variables continuas, se trata de un problema de regresión (Bishop, 2007) por ejemplo, predicción del precio de acciones de bolsa. Por otro lado, cuando el objetivo es predecir una categoría, el problema es de clasificación. Un ejemplo es tener imágenes de vestidos o camisetas y asignarles una etiqueta o bien "camiseta" o bien "vestido". En este trabajo se aborda la clasificación.

Uno de los ejemplos más populares de clasificación es el de clasificación binaria para detectar correos entrantes que sean spam o no. Humanamente no sería posible identificar si el correo es malintencionado o no y gracias a algoritmos de clasificación con ML la máquina es capaz de aprender y procesar las distintas características de los correos de tipo spam y emplearlas como filtro para que todos los correos que entren sean correctamente clasificados antes de llegar al buzón de entrada. En este trabajo se parte de un conjunto de datos previamente etiquetados y con algoritmos de ML para clasificación serán capaces de identificar patrones y aprender.

Clasificador es el término que se emplea para los modelos de aprendizaje de clasificación. Un clasificador se define como una función matemática que mapea datos de entrada en una categoría o clase, es decir, es un sistema que importa un vector de valores de características discretas o continuas y que da como salida un valor único discreto como la clase (Marsland, 2015). Las observaciones hechas se conocen como características, atributos o clases, también etiquetas. Además la clasificación puede ser considerada como dos materias distintas: clasificación binaria o clasificación múltiple. Por ello, las etiquetas pueden ser traducidas a números 0, 1, 2 (James, Witten, Hastie, & Tibshirani, 2013).

Existen múltiples clasificadores con múltiples propósitos, desarrollados para resolver problemas de clasificación. Existen clasificadores basados en reglas como los árboles de decisión que pueden encontrar patrones en datos cualitativos. Además hay otros clasificadores como el *K Nearest Neighbors* y *Adaboost* que predicen tanto datos cuantitativos como cualitativos (James et al., 2013). Cada algoritmo tiene distinto resultado o rendimiento materializado en la precisión de datos y ello depende también de la distribución de los datos y la naturaleza de los mismos.

Gracias a la clasificación se puede a partir de gran cantidad de variables llevar a cabo tareas que de forma humana no sería posible sirviendo de ayuda para la toma de decisiones.

4.4. Aprendizaje automático en Python: la biblioteca Scikit-learn

Existen varios lenguajes de programación que son útiles para aplicar el aprendizaje automático por ejemplo, Python, R, SAS o Java. Todos ofrecen muchas posibilidades para realizar algoritmos de ML. En la actualidad es muy popular Python[4] debido a su peculiaridad con la programación orientada a objetos y la cantidad librerías fáciles e intuitivas facilitan su empleo y aprendizaje. Por ejemplo, las librerías NumPy y SciPy desarrolladas sobre otras capas como Fortran o C para operaciones vectorizadas de gran rendimiento en arrays multidimensionales. Python es un lenguaje de programación que permite modularizar los programas y por lo tanto el código puede ser reusado en futuros trabajos.

En este trabajo se emplean librerías de Python que sirven para el ML, en concreto *Scikit-learn* de libre uso y extensamente usada para ML y minería de datos. Otra librería no utilizada pero también popular es *scikit-image* que contiene algoritmos para el procesado de imágenes. *Scikit-learn* es de simple descarga e instalación, y al tener librerías dependientes de la misma, como Numpy y Scipy, es compatible con distintos sistemas operativos y fácil de usar. Dispone de una API de sencillo uso para poder aplicar distintos módulos que pueden ser importados en cualquier trabajo de Python. Esta API se caracteriza por:

a. Consistencia: Los objetos disponibles comparten una interfaz consistente y simple y se pueden encontrar unos métodos iguales para todos.
b. Clasificadores: Definen mecanismos de instanciación de objetos y donde podemos aplicar el método *fit()* para que aprenda.
c. Predición: interfaz que da la capacidad a un clasificador de añadir el método *predict()* para predecir la salida de cada una basándose en el modelo que ha aprendido en la fase de aprendizaje (*fit*()). Esto se hace en aprendizaje supervisado para hacer una estimación del comportamiento del clasificador para nuevas instancias no vistas previamente. Gracias a estos métodos podemos saber en aprendizaje supervisado la precisión que ha tenido, ya que

[4] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, 131 Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2012. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, (2012), 2825–2830. DOI:https://doi.org/10.1007/s13398-014-0173-7.2

al saber las etiquetas que corresponden, podemos estimar cuántas instancias ha predicho correctamente y cuántas no.

d. Transformadores: A veces es preciso que los datos sean pre-procesados o filtrados para poder pasarlos al clasificador siguiendo una estandarización previa. Un ejemplo es el método *transform()* o *StandardScaler()*
e. Representación de los datos: En *scikit-learn* se representan los datos como un array multi-dimensional haciendo uso del paquete NumPi o el uso de dataframe que nos permite el paquete pandas disponible para Python.
f. No hay cambios de clases: Los conjuntos de datos son representados como arrays de la clase Numpy o matrices de la librería SciPy por lo que las estructuras de datos no son inventadas ni deben ser analizadas para que Scikit-learn pueda procesarlas.
g. Flujo de datos: Se pueden reusar las estructuras de datos pasando como argumento a un clasificador el mismo conjunto de datos transformado varias veces.
h. Instanciación por defecto: Todos los algoritmos disponibles se pueden usar sin tener que configurar los parámetros a la hora de usarlos, los mismos vienen configurados con unos valores por defecto.

4.5. Clasificación y evaluación: Métricas de un algoritmo

Existen varias medidas para evaluar el rendimiento de un algoritmo de clasificación además de la precisión para decidir cuál ofrece mejor resultado al objetivo planteado. No obstante, es necesario comprender qué es la matriz de confusión, herramienta de gran utilidad para evaluar un algoritmo. Se trata de matriz que muestra en un cuadro distintos valores que se obtienen tras realizar el entrenamiento y la validación posterior con el conjunto de datos tipo test usando la validación cruzada: falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos. He aquí un ejemplo para un problema de clasificación binaria:

Ilustración 3. Matriz de confusión

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos(FP)	Verdaderos Negativos(VN)

Fuente: elaboración propia

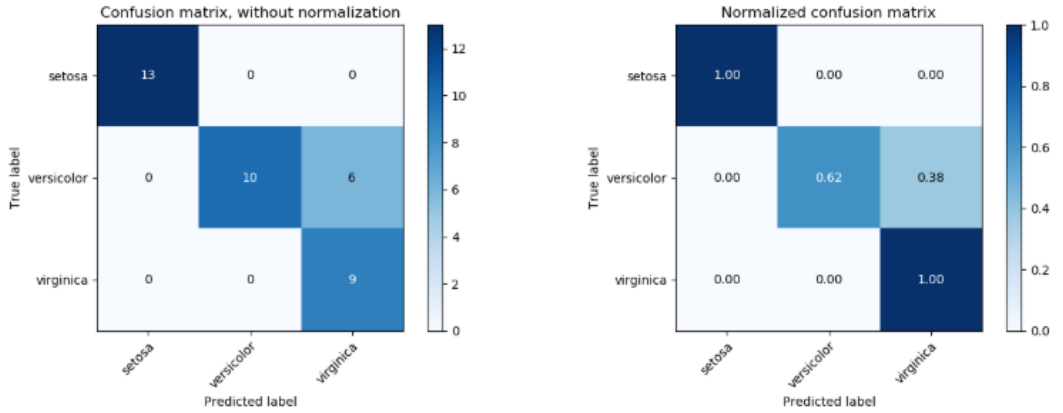
Las etiquetas que son conocidas se representan en las filas y las que el algoritmo predice se representan en las columnas. Estos valores son:

- **Verdaderos Positivos (VP):** la cantidad de aquellas observaciones que se clasificaron como pertenecientes a una clase y acertó.

- **Falsos Positivos (FP):** observaciones que no pertenecían a una clase pero se consideraron perteneciente a ellas, son considerados positivos pero al conocerse la salida sabemos que son equivocaciones.
- **Falsos Negativos (FN):** observaciones que pertenecían a una clase pero no se consideraron en ella, considerados como negativos.
- **Verdaderos Negativos (VN):** aquellas observaciones que no pertenecían a una clase y las clasificó correctamente.

Con el método método *confusion_matrix* del paquete *sklearn.metrics* se obtiene la matriz de confusión para cualquier algoritmo que se entrene y obtener estos cuatro valores que son usados para calcular métricas. En los elementos de la diagonal se representa el número de puntos en los que la etiqueta predicha es igual a la etiqueta verdadera mientras que en los elementos fuera de la diagonal se muestran los elementos mal clasificados por el clasificador. Cuantos más altos sean los valores de la diagonal de la matriz de confusión mejor, indica un mayor número de predicciones correctas. La siguiente figura muestra la matriz de confusión con y sin normalización por el tamaño de la clase. Este tipo de normalización puede ser interesante en el caso de clases no balanceadas para tener una mejor interpretación visual de qué clase está siendo mal clasificada.

Ilustración 4. Matriz de confusión sin y con normalización



https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

Estas cuatro métricas descritas son para evaluar problemas de clasificaciones binarias. Dado que el problema de este trabajo es una clasificación multiclase es preciso obtener el valor promedio para cada una, así para cada métrica tenemos VPi que son los Verdaderos Positivos para la clase i, VNi = Verdaderos Negativos para la clase i, FPi= Falsos Positivos para la clase i y por último FNi = Falsos Negativos para la clase i. Las métricas que se usan en este trabajo son las siguientes:

- ✓ **Precisión:** mide la habilidad de un clasificador de no etiquetar como positivo una observación que se debe considerar como negativa. Cuanto mayor sea este valor mejor será.

$Precision = \frac{VP}{VP + FP}$	Precision promedio $= \frac{1}{C}\sum_{i=1}^{C}\frac{VP_i}{VP_i + FP_i}$

Una forma trivial de tener una precisión perfecta es hacer una sola predicción posible para asegurarse de que es correcta (precisión=1/1=100%). No sería muy útil porque el clasificador ignoraría todos y cada uno de los casos positivos. Por eso se utilizan otras medidas.

- ✓ **Sensibilidad o *Recall***: es la tasa de verdaderos positivos y mide la capacidad del clasificador de encontrar todas las observaciones positivas, este valor ha de ser lo mayor posible en una escala del 0 al 100.

Sensibilidad $= \frac{VP}{(VP + FN)}$	Sensibilidad promedio $= \frac{1}{C}\sum_{i=1}^{C}\frac{VP_i}{VP_i+FN_i}$

- ✓ **Exactitud:** permite obtener el porcentaje de observaciones que ha clasificado correctamente, cuanto más cercano sea su valor a 1 mejor será.

Exactitud $= \frac{VP + VN}{VP + VN + FP + FN}$	Exactitud promedio $= \frac{1}{C}\sum_{i=1}^{C}\frac{VP_i + VN_i}{VP_i + FP_i + VN_i + FN_i}$

- ✓ **F1-score:** además de estas medidas aparece el f1-score que combina las anteriores y es la media armónica de la precisión y el recall. Dado que la media simple trata todos los valores iguales, la media armónica otorga más peso a los valores bajos. Como resultado el clasificador solo arrojará un valor alto de f1 si tanto recall como precisión son elevados.

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

4.6. Tipos de Aprendizaje automático

La biblioteca Scikit-learn contiene un amplio abanico para tratar, procesar y clasificar datos ya sean previamente observados o no. En este trabajo nos focalizamos en la clasificación basada en conjuntos de datos previamente observados, aprendizaje supervisado, identificando a qué clase pertenece un objeto nuevo que no hemos observado previamente.

Hay diferentes tipos de algoritmos según el tipo de clasificación que realicen y podemos encontrar:

- Linear Models/ Modelos Lineales
  - Logistic Regression/ Regresión Logística
  - Perceptron / Perceptrón

- Discriminant Analysis/ Análisis Discriminante
  - Linear Discriminant Analysis / Análisis Discriminante Lineal
  - Quadratic Discriminant Analysis / Análisis Discriminante Cuadrático

- Support Vector Machines/ Máquinas de Vectores de Soporte
  - Linear Support Vector Machines / SVM Lineales
  - Polynomial Support Vector Machines / SVM Polinomial

- Neighbors/ Vecinos
  - Kneighbors

- NaiveBayes
  - Bernouilli NB
  - Gaussian NB
  - Multinomial NB

- Trees/ Arboles
  - DecisionTree / Arboles de decisión

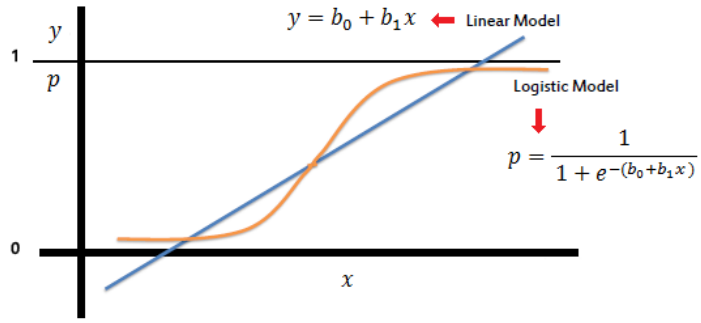
- Ensemble / Ensamblado
  - AdaBoost
  - RandomForest

- Neural Network / Redes Neuronales
  - Multi Layer Perceptron / Perceptrón multicapa

Se describe a continuación brevemente los utilizados en este trabajo:

Regresión Logística: Si bien el nombre de este algoritmo sugiere regresión se trata de un algoritmo de clasificación. Se trata de un clasificador lineal binario que utiliza la técnica OvR (*One versus Rest*). Estadísticamente es un modelo de regresión en el que la variable dependiente es categórica siendo las clases objetivos a predecir y las variables independientes características del conjunto de datos que es usado para entrenamiento.

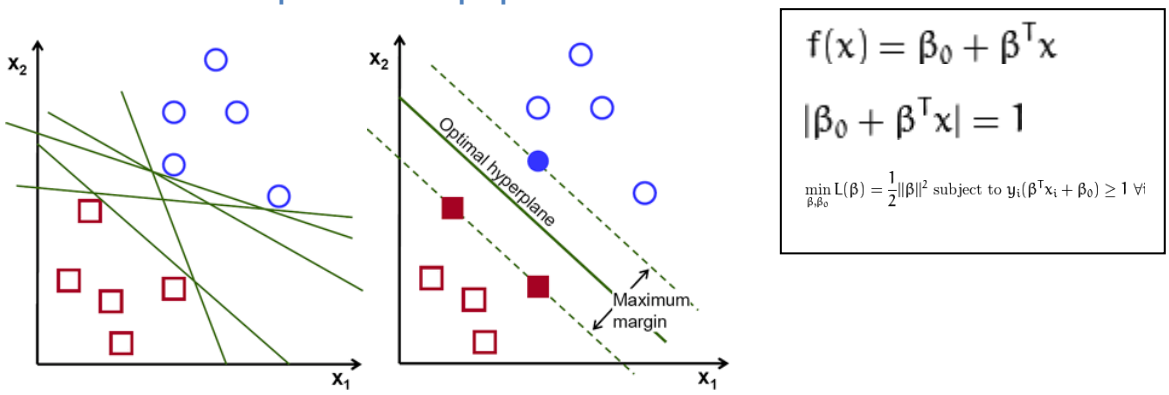
Ilustración 5. Comparación regresión lineal y logística



https://www.saedsayad.com/logistic_regression.htm

Support Vector Machine (SVM): Este algoritmo surgió como competidor más simple de la red neuronal y se conoce de forma abreviada como SVM. Es considerado como una ampliación del Perceptrón y su característica es que logra el hiperplano más óptimo que separa los casos que pertenecen a cada clase en una clasificación binaria. La misión del algoritmo SVM consiste en encontrar el hiperplano que de la distancia mínima más grande a los puntos del conjunto de entrenamiento. Esa distancia se conoce como "margen" y el hiperplano encontrado busca maximizarlo como se aprecia en la siguiente figura[5]:

Ilustración 6. SVM: separación de hiperplano



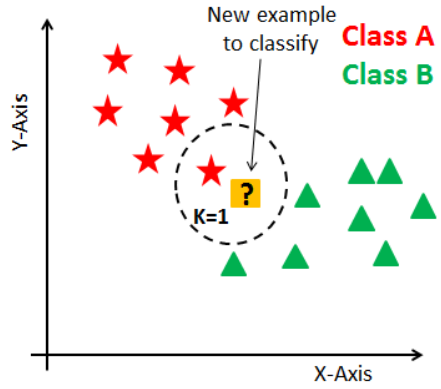
https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

K-Neighbors: El algoritmo K-Nearest Neighbors, llamado KNN, es uno de los más usados y se aplica a distintos campos como el financiero o el sanitario. Permite reconocer patrones y correlaciones y es sencillo de implementar y en especial cuando se desea tener una primera aproximación a un problema de clasificación en el que no se tiene conocimiento sobre cómo se distribuyen los datos ya que no asume cómo están distribuidos los datos (no es paramétrico). Además casi no usa los datos de entrenamiento para construir un modelo generalizado y utilizarlo para clasificar, esos datos son más bien usados en la fase de test por eso la comunidad se refiere a este algoritmo como *lazy* o perezoso. La clasificación se basa en la búsqueda de características similares con cada uno de los datos que se tienen en el conjunto de entrenamiento, esto implica que requiere mayor almacenamiento en memoria al tener

5 Para mayor conocimiento sobre separación de hiperplanos revisar la sección 4.5 del libro *Elements of Statistical Learning* by T. Hastie, R. Tibshirani and J. H. Friedman.

que usar este conjunto en su totalidad para comparar uno por uno. Como es no paramétrico supone que el mejor modelo de los datos son los mismos datos. La clasificación para cada nuevo objeto observado consiste en la asignación a la clase que obtenga mayor votación de sus k vecinos más cercanos. El valor k es número de vecinos más cercanos con características similares para realizar la comparación y proceder a la elección. A pesar de las ventajas de implementación tiene como inconveniente que es muy sensible a datos no relevantes, a la dimensionalidad de los datos, al ruido y además lento si el conjunto de datos de entrenamiento es de gran tamaño. La siguiente figura muestra el algoritmo[6]:

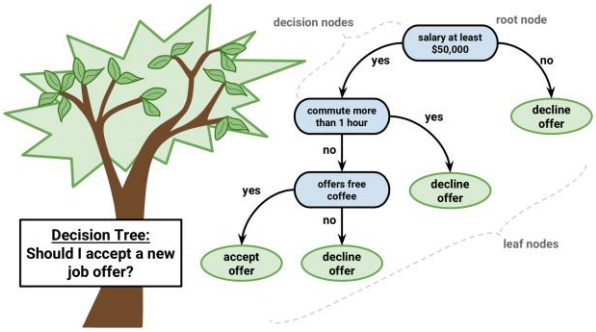
Ilustración 7. K-Neighbors



https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-lear

Arboles de decisión: Una ventaja de este algoritmo usado también en distintos campos es la visualización de su efectividad gracias a sus reglas de clasificación. Los árboles de decisión se definen como un proceso que de forma recursiva particiona o agrupa los datos según unas reglas que son definidas en cada nodo del árbol o rama. Cada nodo de por sí tiene una regla de decisión que determina qué instancias son asignados a cada nodo hijo. Además, la etiqueta de clase del nodo hoja será la clase predicha por el modelo de aprendizaje (Friedl & Brodley, 1997) como se aprecia en esta figura:

Ilustración 8. Arboles de decisión ejemplo



https://www.datacamp.com/community/tutorials/decision-trees

[6] Puede consultarse en detalle N. Suguna and K. Thanushkodi. 2010. An Improved k-Nearest Neighbor Classification Using Genetic Algorithm. Int. J. Comput. Sci. Issues 7, 4 (2010), 18–21.

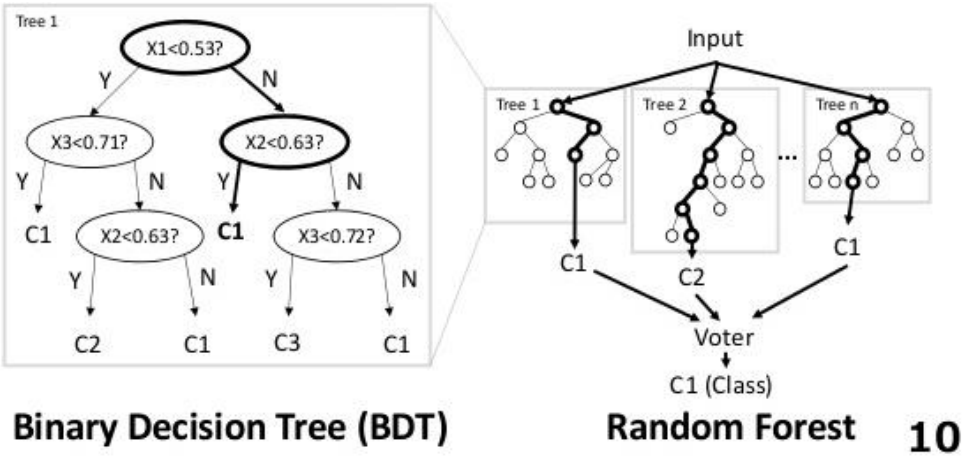
Se han desarrollado distintos tipos de árboles de decisión a lo largo del tiempo. El más popular es el algoritmo ID3 (*Iterative Dichotomiser 3*) de Ross Quinlan en 1986[7]. Este algoritmo crea un árbol que arroja la mayor ganancia de información para objetivos categóricas. Los árboles se construyen al mayor tamaño y luego se podan para aumentar la capacidad predictiva. Son evoluciones de ID3 los algoritmos C4.5 que elimina la restricción de que los atributos deban ser categóricos y la última versión de ID3 es el C5.0 utiliza menos memoria y construye árboles más precisos[8]. Otro tipo parecido al C4.5 es el CART (*Classification and Regression Trees*) que sirve para usar con variables numéricas como objetivo y por lo tanto se puede usar para regresión. Este algoritmo construye árboles binarios dejando en el nodo para decisión el atributo que tenga la mayor ganancia de información. La biblioteca Scikit-learn utiliza el algoritmo CART optimizado.[9] Una desventaja de este algoritmo es que en comparación a otros métodos de ML puede arrojar menor precisión si los datos son uniformes siendo posible que exista sobreajuste cuando no se podan las ramas en las que debe parar y por lo tanto es posible que de un mayor error.

Modelos de aprendizaje de ensamblado: Estos modelos utilizan múltiples algoritmos de aprendizaje para obtener mejor resultado predictivo que el que puede obtenerse por un solo modelo de aprendizaje (Bishop, 2007). Hay dos tipos dependiendo de la técnica empleada[10]:

Bagging o Boostrap aggregating: Se crean diferentes clasificadores independientes y luego se realiza un promedio de sus predicciones. Utilizar el promedio es mejor que usar la predicción de uno solo ya que la varianza se reduce. Un ejemplo de *bagging* es *Random forest*[11] que construye muchos árboles de decisión Cada árbol de decisión que se crea se realiza a partir de una secuencia aleatoria de un subconjunto de datos del conjunto de entrenamiento. La diferencia con un árbol de decisión es que en el nodo donde se realiza la división de los datos no toma el atributo con mayor ganancia o índice gini, adopta la mejor división entre un subconjunto aleatorio de todo el conjunto de características. Esto provocará una desviación mayor del árbol pero al usar la media de varios árboles la varianza disminuye. Este algoritmo clasifica así: dado un ejemplo con sus respectivas características se sitúa en cada uno de los árboles de decisión que hay en el bosque y que el algoritmo ha creado. Cada árbol otorga una clasificación y ese árbol le da una votación a esa clase que ha predicho. El bosque elige la clasificación a través de aquella clase que haya sido más votada como muestra esta figura:[12]

[7] Quinlan, R.L. (1986). Induction of Decision Trees. Mach. Learn. 1, 1 (1986), 81–106.
8 Is C5.0 Better Than C4.5? Recuperado de http://rulequest.com/see5- comparison.html.
9 Tree algorithms: ID3, C4.5, C5.0 and CART. Recuperado de http://scikitlearn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart.
[10] Ensemble methods. Recuperado de: http://scikitlearn.org/stable/modules/ensemble.html#ensemble-methods
[11] RandomForests.http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearnensemble-randomforestclassifier
[12] La implementación que scikit-learn hace de esto es la combinación de varios clasificadores basándose en su predicción probabilística, en lugar de usar la votación que cada clasificador le dé a la clase del ejemplo clasificado.

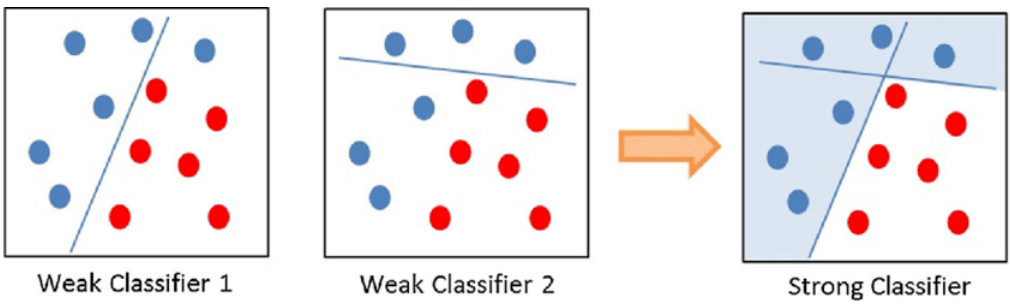
Ilustración 9. Random Forest frente a árbol de decisión binario



https://www.slideshare.net/HirokiNakahara1/a-random-forest-using-a-multivalued-decision-diagram-on-an-fpga

Boosting: Se van creando clasificadores de forma secuencial intentando corregir la dirección del anterior atendiendo al error cometido en sus reglas. La idea es construir un clasificador más preciso partiendo de varios con menor precisión o ajuste. Un ejemplo es el modelo AdaBoost (Adaptative Boosting, AB) creado por Freund y Schapire. Éste método primero analiza un número de algoritmos débiles, que serán árboles de decisión pequeños de profundidad, con poca precisión a través del conjunto de entrenamiento modificado en cada iteración. Secuencialmente construye uno nuevo más preciso con la combinación de todos ellos asignando un peso en la votación final para la elección de las mejores reglas de cada uno. [13]

Ilustración 10. AdaBoost



https://www.researchgate.net/figure/Concept-of-AdaBoost_fig5_264713074

.

[13] Para profundizar Robert E. Schapire. 2013. Explaining adaboost. Empir. Inference Festschrift Honor Vladimir N. Vapnik (2013), 37–52. DOI:https://doi.org/10.1007/978-3-642-41136-6_5

PARTE II. INVESTIGACION EMPIRICA Y RESULTADOS

CAPITULO V. EL CONJUNTO DE DATOS

En un problema de ML de clasificación hay varios retos a superar. El primero es el conjunto de datos del que se dispone y que se van a tratar ya que pueden no tener calidad suficiente debido a que exista un elevado número de datos perdidos, pocos datos para el entrenamiento o datos que no sean relevantes y no sean valiosos como información. El segundo reto es elegir un algoritmo adecuado para dar respuesta al problema de clasificación. En el presente capítulo se muestra el primer reto, el conjunto de datos de que se dispone así como los distintos algoritmos de clasificación que pueden aplicarse.

5.1. Elección del conjunto de datos

Para poder llevar a cabo ML se necesita un número grande de datos para poder entrenar distintos algoritmos. Además se requiere que los datos que se tienen en el conjunto de datos de entrenamiento sean lo suficientemente representativos para el conjunto de datos tipo test, es decir, que el modelo obtenido a través del conjunto de entrenamiento sea válido para poder clasificar adecuadamente nuevos datos que no hayan sido observados previamente. Si el modelo que se ha entrenado sigue una estructura lineal pero el nuevo dato a clasificar no tiene ninguna relación con los datos de entrenamiento, entonces no se logrará precisión en la identificación y clasificación del dato. También es necesario que los datos para entrenar tengan una calidad aceptable.

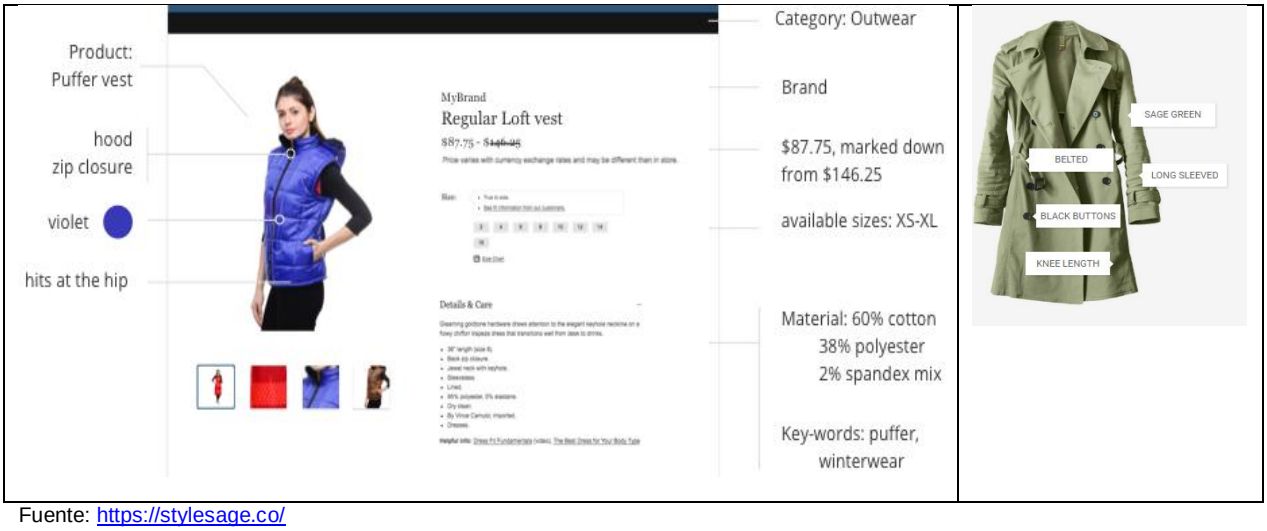
Otro elemento importante es que los datos con los que se entrena el algoritmo tengan atributos que sean relevantes para extraer un modelo de ellos. Puede que los datos tengan gran cantidad de atributos o características pero algunas de ellas introducen ruido o datos que no interesan para obtenerlo, haciendo que aumente el coste de cómputo al tener que realizar mayor cantidad de cálculos. Una técnica para amortiguar este problema es la extracción de atributos o la selección de aquellos que permiten obtener rendimientos similares y, como consecuencia, menor coste de computación como el uso de memoria como el caso del PCA (*Principal Component Analysis*).

5.2. El conjunto de datos a tratar

La base de datos ha sido facilitada por la *start-up* estadounidense de *Big Data*. *https://stylesage.co/.* Antes de proceder a la descripción de las variables y sus datos, es necesario hacer referencia a la empresa fuente de creación de los datos.

StyleSage es una compañía tecnológica de nueva creación con sede en Estados Unidos y en proceso de expansión. La compañía opera en el sector de datos de fabricantes y distribuidores de moda. Es propietaria de una tecnología, siendo su ventaja competitiva, que utiliza reconocimiento de imagen y ML para determinar con total precisión cualquier detalle en el diseño de una prenda de ropa como el estampado, el color, la talla, la silueta y el material, elementos básicos para el éxito de la gama de productos. Esta información es un activo de gran valor para una industria tan dinámica como la moda. Su herramienta "ve y piensa" como un humano para capturar de forma automática estos detalles. Gracias a la experiencia, la máquina es virtualmente más inteligente y trabaja mejor y más rápido logrando recoger millones de puntos de datos de comercio electrónico todos los días y en tiempo real. Gracias a la inteligencia artificial, recogen los datos virtualmente en 360 grados y la información que se puede visualizar en la página de cualquier *retailer* de moda del mundo se recoge automáticamente en una base de datos:

Ilustración 11. StyleSage: origen de los datos



Fuente: https://stylesage.co/

Estos datos son de gran utilidad para los equipos de estrategia, planificación, marketing, merchandising, aprovisionamiento y los equipos de diseño para elaborar surtidos que sean fácilmente vendidos, reaccionen de forma rápida a los cambios del mercado y consigan atraer y retener a los clientes.

El tipo de preguntas que pueden hacerse las empresas a partir de estos datos son las siguientes:

a. ¿Qué tipo de materiales y perfiles lanzan los competidores?
b. ¿Qué estilos están más en rebajas ahora
c. ¿Qué sobrecoste hay entre niño y adulto?
d. ¿Qué tallas y materiales se están vendiendo?

El conjunto de datos contiene tanto datos estructurados, semiestructurados y no estructurados. Para poder dar respuesta a los objetivos de investigación definidos al inicio del proyecto con *StyleSage* fueron necesarias varias fases y discusiones sobre la muestra necesaria. Debido a la dimensión de los datos se tomó la decisión de centrarse en una sola categoría de ropa, los vestidos y en un solo género, vestidos de

mujer y de niña. Otra condición importante que se impuso a la muestra fue incluir un número representativo de países, de *retailers* y de marcas. Así como abarcar un histórico importante para poder modelizar los datos.

Se lograron un total de 428.600 casos, tamaño suficiente para llevar a cabo procesos de ML y un total de 28 atributos. No todos los atributos son de tipo numérico, por ejemplo *title* es un texto. También hay variables categóricas.

Tabla 1. Atributos del conjunto de datos

• Identidad del producto • Nombre de la tienda web • Código del país • Nombre del país • Marca • Título • Texto descriptivo • Código de idioma • Título traducido • Descripción traducida • Primera vez visto • Ultima actualización • Disponible • Nombre Nivel 1	• Nombre Nivel 2 • Nombre de género • Mix de material • Color del retailer • Color del producto 1 • Color del producto 2 • Base de color del producto • URL del producto • Precio original • Precio actual • Código de moneda • Número de tallas • Lista de tallas del retailer • Lista de tallas normalizada

Fuente: elaboración propia

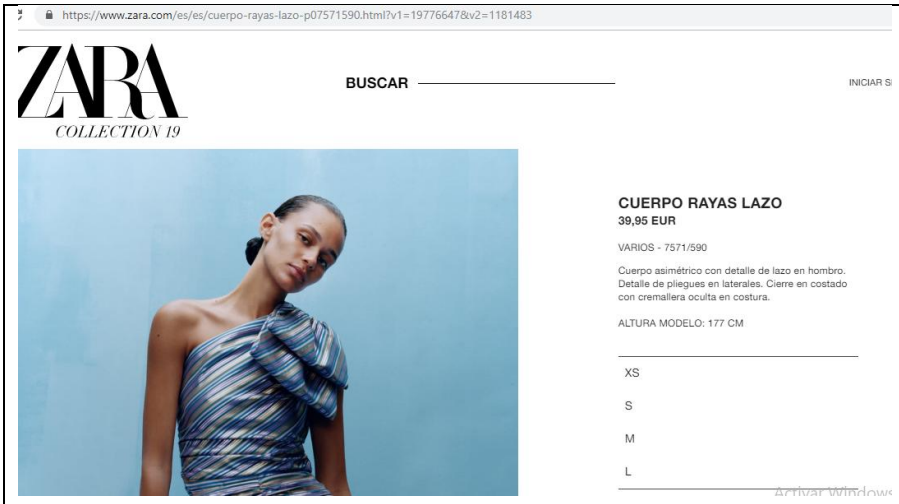
De estos atributos con el fin de explicar el conjunto de datos es necesario comprender:

- Sobre la columna nombre de la página web: hay tiendas webs multimarca, por ejemplo Zalando, Zalando es el nombre de la página web.
- Sobre marca y relacionado con la anterior: Zara es la tienda web pero puede contener varias marcas como Zara Woman o Zara Kids.
- Título: es como se llama la prenda, ejemplo: vestido “paraíso”. Este título es traducido al inglés en una columna siguiente denominada “título traducido”.
- Texto descriptivo: contiene toda la descripción de atributos de la prenda como el nombre, el material, la talla, cada retailer usa su texto para describir la prenda. Este texto es posteriormente traducido en una columna denominada “descripción traducida” al inglés.
- Código de idioma: la base contiene países e idiomas distintos.
- Primera vez visto: registra la fecha en la que la prenda es detectada por primera vez por el algoritmo de StyleSage. Se registra en día, mes y año.
- Ultima actualización: última vez que se ve la prenda y registra un cambio.
- Disponible: si la prenda puede servirse o no en la compra online (“stock virtual”).
- Nombre nivel 1 y 2: son variables que clasifican de ropa a vestidos.
- Nombre género: en este caso es mujer pero se verá que hay varias categorías.
- Mix de material: variable en texto explicativa del material de la prenda.
- Columnas de color: color del retailer se refiere a cómo el retailer nombra el color de la prenda (ej. Rosa flamenco). Color del producto 1 y 2, y base de color del producto.
- Precio original (primer precio al que aparece la prenda).

- Precio actual (precio último que registra la prenda y visto por el algoritmo de la empresa).
- Url del producto: url de dónde sale la información de la prenda.
- Código de moneda: divisa en la que está la prenda según el país.
- Columnas de tallas: número de tallas se refiere a la variedad de tallas, tallas de retailer es cómo las llama el retailer, talla normalizada, se refiere a estandarizadas.

Se muestra el siguiente ejemplo obtenido de la web del fabricante Zara:

Ilustración 12. Atributos del conjunto de datos: ejemplo de Zara



URL de la tienda web y del producto

- ID del producto
- Título: Cuerpo Rayas Lazo
- Precio actual: 39.95
- Moneda: EUR
- Descripción
- Talla: 4 tallas

- Descripción: Cuerpo asimétrico con detalle de lazo en hombro. Detalle de pliegues en laterales. Cierre en costado con cremallera oculta en costura.

Fuente: elaboración propia a partir de www.zaraonline.es

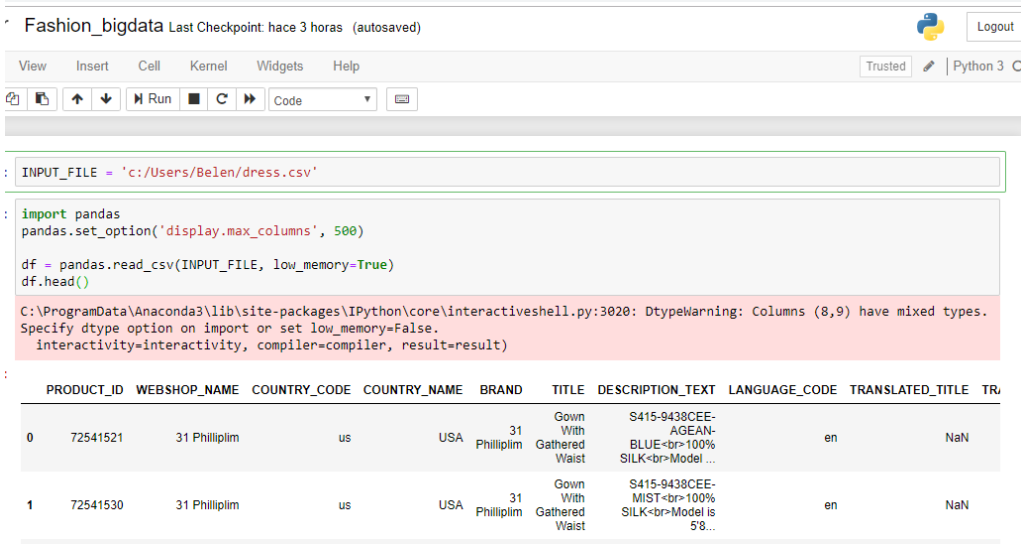
CAPITULO VI. PROCESAMIENTO Y EXTRACCIÓN DEL CONJUNTO DE DATOS

Este capítulo muestra los distintos pasos que se llevaron a cabo para todo el proceso de ML y conseguir el objetivo de clasificación deseado. Como se mencionó previamente se utilizó el lenguaje Python además este trabajo utiliza la biblioteca *scikit-learn* diseñada para su uso en el lenguaje de programación Python. Se ha utilizado la versión 3 de Python. También se han usado otros paquetes para el procesado de los datos, la visualización y tratamiento para operaciones numéricas que scikit-learn necesita para su uso como: *Matplotlib* para visualización de gráficas en *Python, Numpy*: Paquete para computación científica, Pandas: Paquete para el manejo de estructuras de datos, *Scipy*: Basado en Python y usado para operaciones matemáticas, científicas y de ingeniería o *Seaborn*: Paquete basado en matplotlib que sirve para visualización estadística de datos en Python.

6.1. Preprocesado del conjunto de datos

Para poder llevar a cabo un problema de clasificación primero se ha de conocer el conjunto de datos, qué valores tiene, cómo se distribuyen, las correlaciones, si existen datos que tengan que eliminarse, si la naturaleza de los datos es texto o no. Lo primero es cargar la base de datos desde el fichero CSV delimitado por comas. Para ello se utilizó la librería Pandas construida sobre las bases de NumPy. Esta librería comenzó a ser desarrollada en el año 2008 por Wes McKinney (McKinney, 2012) que perseguía encontrar una herramienta flexible y de rendimiento para datos financieros. En la actualidad Pandas constituye un proyecto participado por varios desarrolladores y es una de las librerías más importantes de Python utilizada en el análisis preliminar de los datos. Usando el método *head*() de DataFrame se observa que cada fila representa un vestido y que hay un total de 28 atributos o columnas: product_id, webshop_name, country_code, brand, title… Con el método *info*() se obtienen una descripción general de los datos:

Ilustración 13. Estructura de los datos



	PRODUCT_ID	WEBSHOP_NAME	COUNTRY_CODE	COUNTRY_NAME	BRAND	TITLE	DESCRIPTION_TEXT	LANGUAGE_CODE	TRANSLATED_TITLE	TR
0	72541521	31 Phillplim	us	USA	31 Phillplim	Gown With Gathered Waist	S415-9438CEE-AGEAN-BLUE 100% SILK Model ...	en	NaN	
1	72541530	31 Phillplim	us	USA	31 Phillplim	Gown With Gathered Waist	S415-9438CEE-MIST 100% SILK Model is 5'8...	en	NaN	

Fuente: elaboración propia a partir del editor Jupyter.

Para proceder con el preprocesado de los datos se separó el análisis en análisis univariante y bivariado respecto a la variable objetivo que se define posteriormente.

6.2. Exploración univariante del conjunto de datos.

En este apartado y con el propósito de ofrecer una descripción clarificadora de los datos de partida se muestra la exploración univariante del conjunto de datos siguiendo el orden de las columnas del data frame.

Primero se sospecha que una prenda pueda repetirse en varios países mediante el atributo *Product_ID*. Se busca el número de identificadores distintos en el total y se observan 270.963 de los 428.600 del conjunto total.

Identificador de producto

```
print('Number of different items {0}'.format(len(df.PRODUCT_ID.unique())))
print('Number of rows {0}'.format(len(df)))
```

```
Number of different items 270963
Number of rows 428600
```

	PRODUCT_ID	freq
176171	234009464	18
1941	52034401	18
1968	52034518	18
1969	52034519	18
1970	52034520	18

Se obtiene un caso aislado, la prenda con ID de producto 234009464 comprobando que se encuentra en varios países.

Product: 234009464

```
idp = 234009464
df[df.PRODUCT_ID == idp]
```

	PRODUCT_ID	WEBSHOP_NAME	COUNTRY_CODE	COUNTRY_NAME	BRAND	TITLE	DESCRIPTION_TEXT
242438	234009464	Net-A-Porter	ae	United Arab Emirates	Hatch	Luella off-the-shoulder stretch-jersey maxi dress	SIZE & FIT INFORMATION - Size 0 (Petite) = ...
244662	234009464	Net-A-Porter	ar	Argentina	Hatch	Luella off-the-shoulder stretch-jersey maxi dress	SIZE & FIT INFORMATION - Size 0 (Petite) = ...
247070	234009464	Net-A-Porter	au	Australia	Hatch	Luella off-the-shoulder stretch-	SIZE & FIT INFORMATION -

El siguiente atributo que se explora con *value.counts*() es *Webshop_Name* (nombre de la tienda online) y *Brand* (marca). Se trata de variables categóricas. En total hay 267 tiendas online (retailers, fabricantes o distribuidores) y 6.633 nombres de marcas de vestidos. Se apunta que hay bastantes categorías escasamente representadas que deben de tenerse en cuenta a la hora de modelizar.

```
vestidos['WEBSHOP_NAME'].value_count
```

Asos	74902
Net-A-Porter	45033
House of Fraser	18819
Nordstrom	16781
Nordstrom Rack	15401
Walmart	15254
Sears	13440
Neiman Marcus	11120
OutNet	10514
Zalando	10106
MarkaVIP	9586
Saks Fifth Avenue	8626
Zozo	8311
Lord & Taylor	7352
MODA OPERANDI	7107
Mango	6977
Forever21	6952
Macy's	6420
Dillards	5923
H&M	5483
Namshi	5435
Bloomingdale's	5177
Dafiti	4782
JCPenney	4728
Mango Outlet	4548
Zappos	4256
6pm.com	4191
Matches Fashion	4167
T.J. Maxx	4107
Kohl's	3249
...	
Billabong	4
Fay	4
Brooks Brothers	4

```
vestidos['BRAND'].value_counts()
```

Asos Private Labels	18557
Mango	11849
Little Mistress	7923
Adrianna Papell	6220
Forever21	6111
H&M	5084
Tfnc	4795
Unknown	4353
Halston Heritage	3694
Club L	3488
Jarlo	3427
Maya	3289
Lauren Ralph Lauren	2973
Needle & Thread	2733
Designer	2506
Badgley Mischka	2457
Zara	2381
Forever Unique	2351
Bcbg Maxazria	2242
Forever 21	2144
Xscape	2131
Tadashi Shoji	2082
Lipsy	2073
Alice & Olivia	2007
Aidan Mattox	1929
Free People	1870
Diane Von Furstenberg	1866
Oscar De La Renta	1841
Marchesa Notte	1796
Michael Kors	1781
...	
B Slim	1
Paradis Terre	1

Del mismo modo se obtienen las variables código país y nombre de país que se sospechan repetida por lo que se comparan las frecuencias y se obtiene que estas variables son categóricas, 18 clases en total y que hay una limitación importante en la muestra: el 50% de los casos viene representado por dos países: Estados Unidos e Inglaterra significando un sesgo relevante que afecta a las decisiones de modelización.

COUNTRY_CODE	COUNTRY_NAME
us	USA
gb	UK
de	Germany
nl	Netherlands
fr	France
au	Australia
es	Spain
it	Italy
jp	Japan
ru	Russia
sa	Saudi Arabia
ae	United Arab Emirates
br	Brazil
tr	Turkey
mx	Mexico
cn	China
in	India
ar	Argentina

A continuación, se estudian los atributos *language_code* (código de lenguaje) y *translated_title* (título traducido). Se busca si hay alguna traducción no disponible identificándose a título anecdótico que las prendas en portugués no están traducidas.

LANGUAGE_CODE y TRANSLATED_TITLE

```
df.loc[~df.TRANSLATED_TITLE.isnull(),['TRANSLATED_TITLE','TITLE','LANGUAGE_CODE']].head()
```

	TRANSLATED_TITLE	TITLE	LANGUAGE_CODE
4251	infant christening gown	Baby Taufkleid	de
4252	dress without sleeve 'rabine'	Kleid ohne Ärmel 'Rabine'	de
4253	maxi 'nitjisola'	Maxikleid 'nitjisola'	de
4254	maxi 'nitjisola'	Maxikleid 'nitjisola'	de
4255	wings sleeveless gown	Flügelärmel-Kleid	de

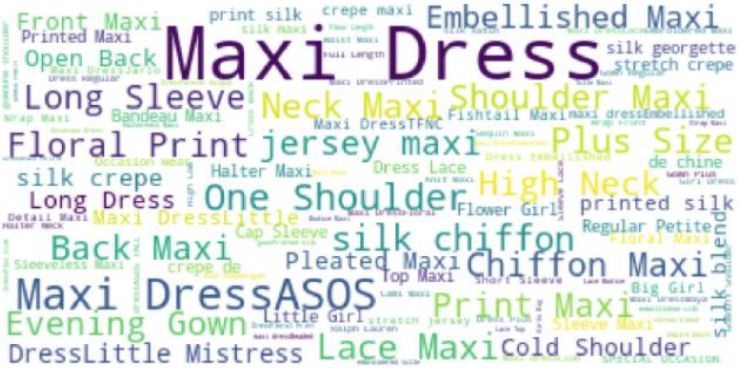
```
print('Las traducciones no están disponibles para los lenguajes:')
df[df.TRANSLATED_TITLE.isnull()].LANGUAGE_CODE.unique()
```

```
Las traducciones no están disponibles para los lenguajes:

array(['en', 'pt', 'EN'], dtype=object)
```

```
df.loc[(df.LANGUAGE_CODE == 'pt')&(df.TRANSLATED_TITLE.isnull()),['TRANSLATED_TITLE','TITLE']].head()
```

	TRANSLATED_TITLE	TITLE
6965	NaN	NaN
6966	NaN	NaN
6967	NaN	NaN
6968	NaN	NaN
6969	NaN	NaN

Se obtiene la frecuencia del atributo *title* mediante un *wordcount* (véase código en Anexo 8) para identificar las palabras más utilizadas a la hora de dar nombre a los vestidos:



A continuación se encuentran las variables *level_name1 y level_name2* (niveles de nombre). Estos atributos cumplen la función de clasificar en *clothing* y *dresses*, lo cual implica variables que no se incluirán directamente en el modelo.

LEVEL1_NAME y LEVEL2_NAME

```
print('Categoría Level1:')
df.LEVEL1_NAME.unique()
```

```
Categoría Level1:

array(['Clothing'], dtype=object)
```

```
print('Categoría Level2:')
df.LEVEL2_NAME.unique()
```

```
Categoría Level2:

array(['Dresses'], dtype=object)
```

A continuación se explora la siguiente variable *Gender_name*. Se sabe a priori que el conjunto de datos como son vestidos son todos de mujer. Se observa que este atributo es categórico conteniendo distintas clases de mujer como mujeres grandes, pequeñas, niñas como se observa a continuación. Nuevamente es necesario destacar el sesgo de la muestra ya que la clase women presenta un 84.8% de los casos y algunas no tienen apenas representación.

GENDER_NAME

```
print('Categoría GENDER_NAME:')
df.GENDER_NAME.unique()
```

```
Categoría GENDER_NAME:

array(['Women', 'Women Plus', 'Women Petite', 'Girls', 'Baby Girls',
       'Women Maternity', 'UNKNOWN', 'Women Tall', 'Men', 'Toddler Girls',
       'Babies', 'Neutral', 'Toddler Boys', 'Unisex', 'Kids'],
      dtype=object)
```

	GENDER_NAME	N
10	Women	84.808213
13	Women Plus	5.693887
12	Women Petite	3.628091
2	Girls	1.960803
14	Women Tall	1.790247
11	Women Maternity	1.463369
1	Baby Girls	0.470369
7	Toddler Girls	0.168689
8	UNKNOWN	0.008633
4	Men	0.003733
0	Babies	0.001167
6	Toddler Boys	0.001167
3	Kids	0.000933
9	Unisex	0.000467
5	Neutral	0.000233

Se muestran los atributos *color product_color1* y *product_color2* (color de la prenda) que vienen en tipo objeto y se procesan de modo hexadecimal a color RGB. Se exploran gráficamente teniendo indicios de que estas variables pueden tener predictor por la forma de la distribución.

PRODUCT_COLOR

```
print('Categoría PRODUCT_COLOR1:')
df.PRODUCT_COLOR1.unique()
```

```
Categoría PRODUCT_COLOR1:
```

```
array(['148cdb', 'd8c9c9', 'e4e3e4', ..., 'eac8cc', 'e19286', 'd6b2ac'],
      dtype=object)
```

```
print('Categoría PRODUCT_COLOR2:')
df.PRODUCT_COLOR2.unique()
```

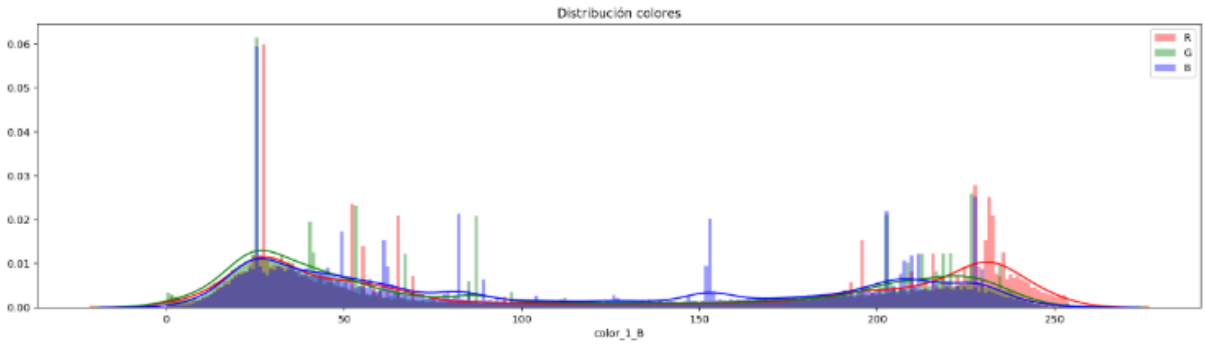
```
Categoría PRODUCT_COLOR2:
```

```
array(['d4b3a5', nan, '6a755d', ..., 'be6d62', 'b58c84', '25233d'],
      dtype=object)
```

Se procesa el tipo de color en hexadecimal a RGB

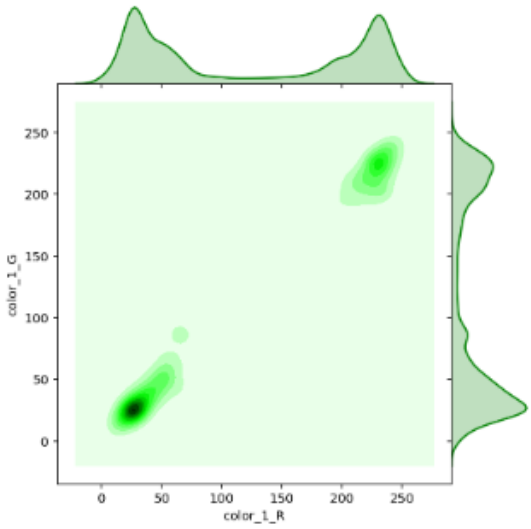
```
import numpy
def hex2rgb(h):
    if type(h) is str and len(h) == 6:
        rgb = tuple(int(h[i:i+2], 16) for i in (0, 2, 4))
        return rgb[0], rgb[1], rgb[2]
    else:
        return numpy.nan, numpy.nan, numpy.nan

df['color_1_R'], df['color_1_G'], df['color_1_B'] = zip(*df['PRODUCT_COLOR1'].map(hex2rgb))
df['color_2_R'], df['color_2_G'], df['color_2_B'] = zip(*df['PRODUCT_COLOR2'].map(hex2rgb))
```

```
fig, ax = plt.subplots(figsize=(20,5))
seaborn.distplot(df.color_1_R.dropna(), bins=256, ax=ax, color='red', label='R')
seaborn.distplot(df.color_1_G.dropna(), bins=256, ax=ax, color='green', label='G')
seaborn.distplot(df.color_1_B.dropna(), bins=256, ax=ax, color='blue', label='B')
plt.legend(loc='best')
plt.title('Distribución colores')
plt.show()
```




```
seaborn.jointplot("color_1_R", "color_1_G", data=df,
                  kind="kde", space=0, color="g")
```

```
<seaborn.axisgrid.JointGrid at 0x1ea2dd05a20>
```



Además de estas dos variables relativas al color se detecta en columnas posteriores otra *retailer_color* observándose que el negro es el color más representado (14%).

	RETAILER_COLOR	N
1752	black	14.145384
14253	navy	5.232447
15293	no_color	3.406485
18053	red	2.849045
3996	blue	2.434431
22133	white	2.395050
13895	multi	2.007081
16874	pink	1.934400
15393	nude	1.635157
11112	ivory	1.226090
9947	grey	1.203626
9772	green	1.032234
4619	blush	0.970112
6800	cream	0.836437
17736	purple	0.800938
19765	silver	0.735488
6498	coral	0.683904
9491	gold	0.658944
5255	burgundy	0.557440
15934	orange	0.515563

Se explora el atributo a continuación *currency_code* que indica la moneda de cambio en la que aparece el precio de la prenda (*atributo original_price y current_price*), se convierten todos los precios en dos columnas (*original y current_price*) a dólares.

CURRENCY_CODE

```
df.CURRENCY_CODE.unique()
```

```
array(['USD', 'GBP', 'EUR', 'ARS', 'AUD', 'CNY', 'INR', 'JPY', 'MXN',
       'RUB', 'BRL', 'AED', 'SAR', 'TRY', 'HKD', 'IDR'], dtype=object)
```

```
change_currency = {'GBP':1.26,
                   'EUR':1.12,
                   'ARS':0.022,
                   'AUD':0.69,
                   'CNY':0.14,
                   'INR':0.014,
                   'JPY':0.0092,
                   'MXN':0.051,
                   'RUB':0.015,
                   'BRL':0.25,
                   'AED':0.27,
                   'SAR':0.27,
                   'TRY':0.17,
                   'HKD':0.13,
                   'IDR':0.000070}
```

```
for k, v in change_currency.items():
    print(k,v)
    df.loc[df.CURRENCY_CODE == k, 'ORIGINAL_PRICE'] = df.loc[df.CURRENCY_CODE == k, 'ORIGINAL_PRICE'] * v
```

```
GBP 1.26
EUR 1.12
ARS 0.022
AUD 0.69
CNY 0.14
INR 0.014
JPY 0.0092
MXN 0.051
RUB 0.015
BRL 0.25
AED 0.27
SAR 0.27
TRY 0.17
HKD 0.13
IDR 7e-05
```

Se detecta que la variable *attributes* es de tipo objeto. Se considera que puede contener información relevante para el modelo predictivo y por ello se identifican los objetos dentro de la variable con un ejemplo y se trocea la variable para extraer información sobre longitud (“length”), cuello (“neckline”), ocasión (“occasion”), patrón (“pattern”), estampado (“printed”), fruncido (“ruffle”), lentejuela (“sequin”), silueta (“silhoutte”), largo de manga (“sleeve_length”).

ATTRIBUTES

```
df.ATTRIBUTES.head()
```

```
0    ,length:dress_length:long:Long Dress:99,neckli...
1    ,length:dress_length:long:Long Dress:99,neckli...
2    ,length:dress_length:long:Long Dress:99,neckli...
3    ,length:dress_length:long:Long Dress:100,neckl...
4    ,length:dress_length:long:Long Dress:100,neckl...
Name: ATTRIBUTES, dtype: object
```

```
example = df.ATTRIBUTES.iloc[0]
example
```

```
',length:dress_length:long:Long Dress:99,neckline:dress_neckline:unknown:Unknown:20,occasion:dress_occasion:evening:Evening Dre
sses:99,pattern:dress_pattern:unknown:Unknown:38,printed:dress_printed:unknown:Unknown:35,ruffles:dress_ruffles:no_ruffles:No R
uffles:84,sequin:dress_sequin:no_sequin:No Sequins:85,silhouette:dress_silhouette:ball_gown:Ball Gown:66,sleeve_length:dress_sl
eeve_length:sleeveless:Sleeveless:77'
```

```
for s in df.ATTRIBUTES.iloc[2345].split(','):
    ss = s.split(':')
    print(ss)
```

```
['']
['length', 'dress_length', 'long', 'Long Dress', '99']
['neckline', 'dress_neckline', 'v_neck', 'V-Neck', '58']
['occasion', 'dress_occasion', 'evening', 'Evening Dresses', '99']
['pattern', 'dress_pattern', 'lace', 'Lace', '40']
['printed', 'dress_printed', 'printed', 'Printed', '72']
['ruffles', 'dress_ruffles', 'no_ruffles', 'No Ruffles', '84']
['sequin', 'dress_sequin', 'no_sequin', 'No Sequins', '85']
['silhouette', 'dress_silhouette', 'ball_gown', 'Ball Gown', '60']
['sleeve_length', 'dress_sleeve_length', 'sleeveless', 'Sleeveless', '77']
```

```
def extract_attributes(s):
    extd = dict()
    for s in s.split(','):
        ss = s.split(':')

        for k in ss:
            k = k.replace('(','').replace(')','').lower().replace(' ','_').replace('-','_')
            k = k.replace('__','_').replace('_/_','_')

            if k != '':
                try:
                    a = int(k)
                except:
                    if k[0] == '_':
                        k = k[1::]

                    extd['att_' + k] = 1

        try:
            extd['attn_' + ss[0]] = int(ss[-1])
        except:
            pass

    return extd
```

```
df = pandas.concat([df, df.ATTRIBUTES.apply(extract_attributes).apply(pandas.Series)], axis=1)
```

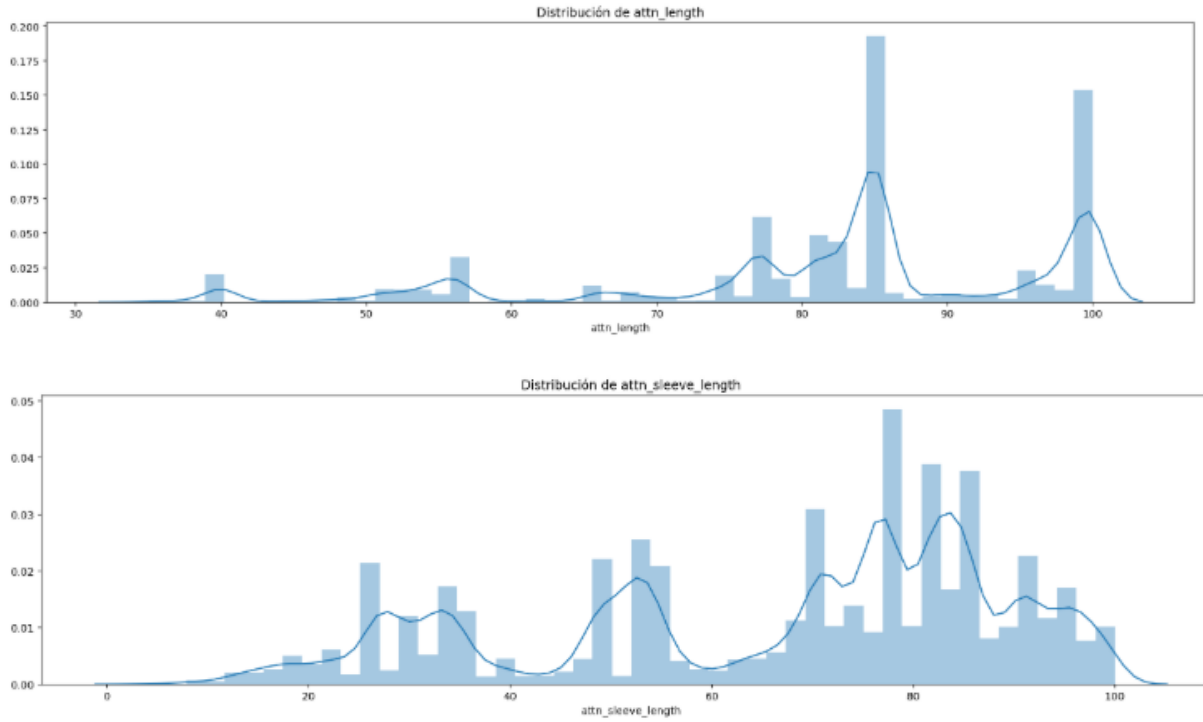
Y se extraen los atributos de la prenda:

```
for c in [c for c in df.columns if 'att_' in c]:
    df.loc[df[c].isnull(),c] = 0
    print(c)
```

```
att_length
att_dress_length
att_long
att_long_dress
att_neckline
att_dress_neckline
att_unknown
att_occasion
att_dress_occasion
att_evening
att_evening_dresses
att_pattern
att_dress_pattern
att_printed
att_dress_printed
att_ruffles
att_dress_ruffles
att_no_ruffles
att_sequin
att_dress_sequin
```

Finalmente se obtiene la distribución de cada atributo cuello, longitud etc, mostrándose para simplificar dos ejemplos (*attn_length* y *attn_sleeve_length*).

```
for c in [c for c in df.columns if 'attn_' in c]:
    fig, ax = plt.subplots(figsize=(20,5))
    seaborn.distplot(df[c].dropna().astype('int'), bins=50, ax=ax, label=c)
    plt.title('Distribución de {0}'.format(c))
    plt.show()
```



Seguidamente aparece el atributo *material_mix* también tipo objeto. Está representando los materiales de los vestidos y es compleja porque las prendas pueden estar compuestas por distintos porcentajes de materiales. Para extraer la información se identifican primero los materiales y luego se extraen cada uno de ellos. Se muestra un ejemplo de prendas compuestas 100% por un solo material.

MATERIAL_MIX

```
df.MATERIAL_MIX.tail()
```

```
428857    Polyester:100
428858    Polyester:100
428859    Polyester:100
428860    Polyester:100
428861    Polyester:100
Name: MATERIAL_MIX, dtype: object
```

```
def extract_materials(s):
    extd = dict()
    for s in s.split(','):
        ss = s.split(':')

        try:
            material = ss[0].replace('(','').replace(')','').lower().replace(' ','_').replace('-','_')
            material = material.replace('__','_')

            extd['material_' + material] = ss[-1]

        except:
            extd['material_unparsed'] = s

    return extd
```

```
df = df[[c for c in df.columns if 'material_' not in c]]
```

```
df = pandas.concat([df, df.MATERIAL_MIX.apply(extract_materials).apply(pandas.Series)], axis=1)
```

```
for c in df.columns:
    if 'material_' in c:
        df.loc[df[c].isnull(),c] = 0
```

```
materialsdf = list()
for c in [c for c in df.columns if 'material_' in c]:
    m = c.replace('material_','')
    p = 100 * (~df[c].isnull()).sum()/len(df)
    materialsdf.append({'material':m, 'percentage':'{0:.2f}%'.format(p), 'pnum':p})

materialsdf = pandas.DataFrame(materialsdf).sort_values('pnum', ascending=False)
materialsdf[['material','percentage']]
```

	material	percentage
0	silk	100.00%
15	faux_leather	100.00%
27	other_wool	100.00%
26	natural_other	100.00%
25	exotic_leather	100.00%
24	chambray	100.00%
23	taffeta	100.00%
22	wool	100.00%
21	denim	100.00%
20	acrylic	100.00%
19	fur	100.00%
18	cashmere	100.00%
17	velvet	100.00%
16	leather	100.00%
14	satin	100.00%
1	crepe	100.00%
13	linen	100.00%
12	nylon	100.00%
11	synthetic_other	100.00%
10	acetate	100.00%
9	metal	100.00%
8	modal	100.00%
7	spandex	100.00%
6	rayon	100.00%
5	unknown	100.00%

La variable a continuación de material_mix es la referente a las tallas de la prenda *size_list_normalized* pero se observa que no es la única en la base de datos. Hay más que hacen referencia a la talla de la prenda: *size_letter* y *size_other*. Se exploran todas y más adelante se unifican. La primera aparece en números, la segunda en letras. Se observa que la talla más representada es la extrapequeña o XS (30%).

SIZE_LIST_NORMALIZED

```
df.SIZE_LIST_NORMALIZED.head()
```

```
0    10,6,2,0,8,4
1    10,6,0,8,2,4
2    4,8,0,6,2,10
3    4,2,10,0,8,6
4      4,8,6,10,2
Name: SIZE_LIST_NORMALIZED, dtype: object
```

```
def extract_sizes(s):
    extd = dict()
    if type(s) is str:
        for s in s.split(','):
            s = s.lower()

            if s != '':

                if s in ['xxs','xs','s','m','l','xl','xxl',
                         '2xl','3xl','4xl','5xl','6xl','7xl','8xl']:
                    if s == 'xxl':
                        s = '2xl'

                    extd['size_letter'] = s
                else:
                    try:
                        ns = int(s)
                        extd['size_num'] = ns
                    except:
                        extd['size_' + s] = 1

                return extd
    else:
        return extd
```

```
df = df[[c for c in df.columns if 'size_' not in c]]
```

```
df = pandas.concat([df, df.SIZE_LIST_NORMALIZED.apply(extract_sizes).apply(pandas.Series)], axis=1)
```

```
for c in [c for c in df.columns if 'size_' in c]:
```

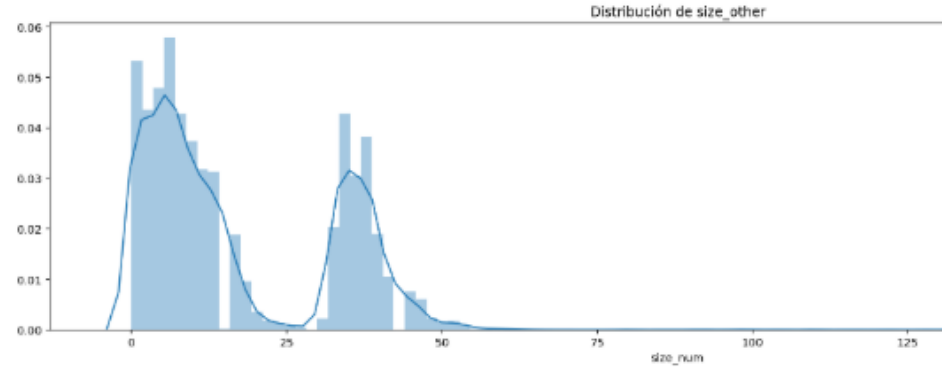
	size_letter	N
11	xs	30.986914
9	s	28.280356
8	m	15.169056
7	l	12.955771
10	xl	6.340763
12	xxs	2.298872
0	2xl	2.064199
1	3xl	1.227660
2	4xl	0.394802
3	5xl	0.190499
4	6xl	0.080985
6	8xl	0.006442
5	7xl	0.003681

Además hay un porcentaje de tallas calificadas como desconocidas (atributo *size_other*) del 11.12% y que sigue una distribución bimodal.

```
print('Unknown sizes {0:.2f}%'.format(100 * df.size_other.sum()/len(df)))
```

```
Unknown sizes 11.12%
```

```
fig, ax = plt.subplots(figsize=(20,5))
seaborn.distplot(df['size_num'].dropna().astype('int'), bins=100, ax=ax, label='size_num')
plt.title('Distribución de {0}'.format(c))
plt.show()
```



Por último hay un atributo denominado ***available*** que indica si en la página web cita si la prenda está disponible o no en la web. Es una variable dicotómica y 73.252 prendas disponibles en la web y 355.346 no disponibles. Importante señalar que esta variable no representa stock físico, es decir si está disponible para vender o no.

AVAILABLE

```
print('Número de muestras con AVAILABLE {0}'.format(len(df[df.AVAILABLE == 1])))
print('Número de muestras NOT AVAILABLE {0}'.format(len(df[df.AVAILABLE == 0])))
```

```
Número de muestras con AVAILABLE 73252
Número de muestras NOT AVAILABLE 355348
```

Tras esta exploración preliminar se toman las primeras decisiones para tratar y preparar el conjunto de datos como se resume en la siguiente figura:

```
## Decisiones 1ªexploración-eliminar columnas
```

```
droped_cols = [
"COUNTRY_NAME", # Ya tenemos var country code que significa lo mismo
"TITLE",
"DESCRIPTION_TEXT",
"LANGUAGE_CODE",
"TRANSLATED_TITLE", # Nube de palabras
"TRANSLATED_DESCRIPTION",
"FIRST_SEEN",
"LAST_UPDATE",
"LEVEL1_NAME", #
"LEVEL2_NAME", #
"MATERIAL_MIX", # Procesada y troceada en materiales que son nuevas variables
"RETAILER_COLOR", # Eliminar porque hay otra variable de color
"PRODUCT_COLOR1", # Cambiada
"PRODUCT_COLOR2", # Cambiada
"PRODUCT_BASE_COLOR", # no usar por ahora en el modelo (es una variable con los matices del color)
"ATTRIBUTES", # troceada y se han creado más variables con atributos
"PRODUCT_URL", # no sirve a no ser que tuviera imagen y pudiera predecir por imagen (siguiente petición)
"CURRENT_PRICE", # usada para variable descuento
"CURRENCY_CODE", # convertido a dólares todo, ya no sirve
"SIZE_COUNT", # Procesada
"SIZE_LIST_RETAILER", # Procesada
"SIZE_LIST_NORMALIZED", # Procesada
"FIRST_SEEN_DATE", # usada para crear días vivos int
"LAST_UPDATE_TIMESTAMP", # idem ambas
"days_alive"]

df = df.drop(columns=droped_cols)
```

```
df.columns = [c.lower() for c in df.columns]
```

```
df.to_csv(OUTPUT_FILE, index=False)
```

Fuente: elaboración propia obtenida del tablero de python

6.3. Creación de variable objetivo y estudio bivariante

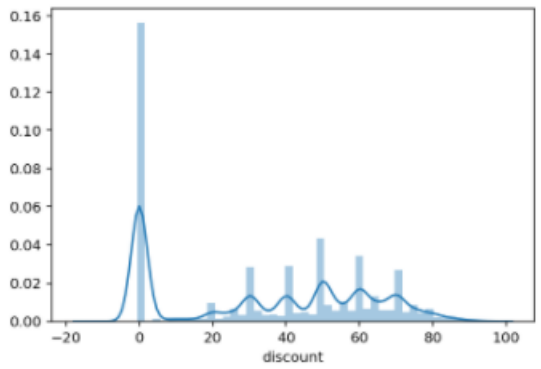
Realizada la exploración univariante de los datos se prosigue creando la variable objeto de estudio, *porcentaje de descuento de la prenda* (a partir del precio original y el precio último) y también se crea una variable que indica los *días vivos de la prenda* dado que se tienen las fechas en día, mes y año de aparición y desaparición de la prenda. Se observa que hay muchas prendas con descuento cero y descuentos en negativo que tendrán que depurarse.

Para crear la variable objetivo porcentaje de descuento del vestido se siguen dos estrategias: a. Tratarla como variable continua (entonces el propósito sería predecirla a través de una regresión lineal) y; b. Categorizarla creando clases de descuento o tramos para cumplir el objetivo de esta investigación, es decir, clasificación.

	product_id	webshop_name	country_code	brand	available	gender_name	original_price	discount	days_alive_int	color_1_r	color_1_g	color_1_b	color_
0	72541521	31 Philliplim	us	31 Philliplim	0	Women	975.0	0.0	113	20.0	140.0	219.0	2
1	72541530	31 Philliplim	us	31 Philliplim	0	Women	975.0	0.0	113	216.0	201.0	201.0	
2	72541532	31 Philliplim	us	31 Philliplim	0	Women	975.0	0.0	113	228.0	227.0	228.0	
3	94171841	31 Philliplim	us	31 Philliplim	0	Women	975.0	40.0	157	228.0	227.0	228.0	
4	94172179	31 Philliplim	us	31 Philliplim	0	Women	975.0	40.0	157	20.0	140.0	219.0	2

```
seaborn.distplot(df.discount)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x108e86828>
```



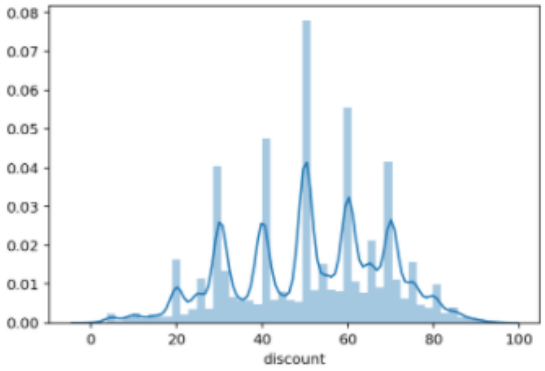

```
: df.discount.describe()
```

```
: count    428600.000000
  mean         34.078793
  std          27.911677
  min         -11.235955
  25%           0.000000
  50%          40.000000
  75%          60.000000
  max          95.000000
  Name: discount, dtype: float64
```

Se obtiene la distribución de las prendas que tienen un descuento superior a 0 para comprender mejor esta variable:

```
seaborn.distplot(df[df.discount>0].discount)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10a46fcc0>
```



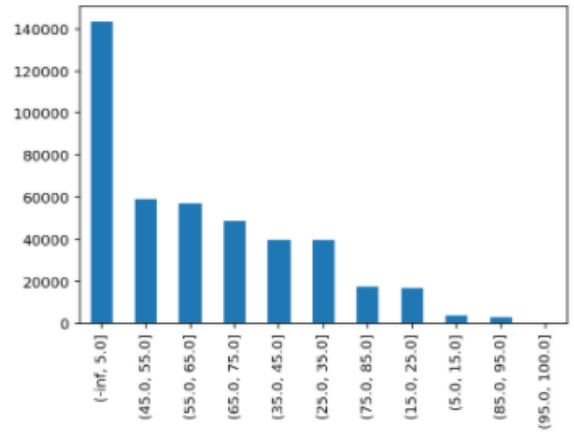
Y se crea la variable categorizada explorando la frecuencia:

```
intervals = pandas.IntervalIndex.from_tuples([(-numpy.inf, 5),
                                              (5, 15),
                                              (15,25),
                                             (25, 35),
                                             (35, 45),
                                             (45, 55),
                                             (55, 65),
                                             (65, 75),
                                             (75, 85),
                                             (85, 95),
                                             (95, 100)])
```

```
df['discount_bin'] = pandas.cut(df.discount, bins=intervals)
```

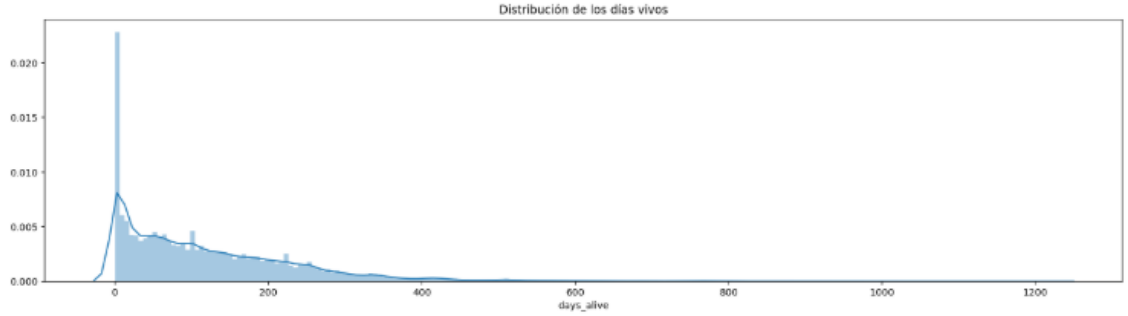
```
df['discount_bin'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10a9a4c18>
```



A continuación se crea la variable días vivos restando la fecha final en días, meses, años menos la inicial obteniéndose la distribución y el porcentaje de prendas con días vivos igual a 0.

```
fig, ax = plt.subplots(figsize=(20,5))
seaborn.distplot(df.days_alive.dt.days, bins=200, ax=ax)
plt.title('Distribución de los días vivos')
plt.show()
```




```
print('Porcentaje de la muestra con días vivos iguales a cero {0}'.format(100 * len(df[df.days_alive.dt.days == 0])/len(df)))
```

```
Porcentaje de la muestra con días vivos iguales a cero 1.960569295380308
```

Variable días vivos

```
# Parse dates
df['FIRST_SEEN_DATE'] = pandas.to_datetime(df.FIRST_SEEN)
df['LAST_UPDATE_TIMESTAMP'] = pandas.to_datetime(df.LAST_UPDATE)
```

```
print('First date of dataset: {0}'.format(df.FIRST_SEEN_DATE.min()))
print('Last date of dataset: {0}'.format(df.LAST_UPDATE_TIMESTAMP.max()))
```

```
First date of dataset: 2013-08-01 00:00:00
Last date of dataset: 2018-01-28 00:00:00
```

```
df['days_alive'] = (df.LAST_UPDATE_TIMESTAMP - df.FIRST_SEEN_DATE)
```

```
print('Min days alive: {0}'.format(df.days_alive.min()))
print('Max days alive: {0}'.format(df.days_alive.max()))
```

```
Min days alive: -29 days +00:00:00
Max days alive: 1222 days 00:00:00
```

```
originN = len(df)
filteredN = len(df[df.days_alive.dt.days >= 0])

print('Dataframe length before filtering: {0}'.format(originN))
print('Dataframe length  after filtering: {0}'.format(filteredN))
print('Total filtered rows: {0}'.format(originN - filteredN))
```

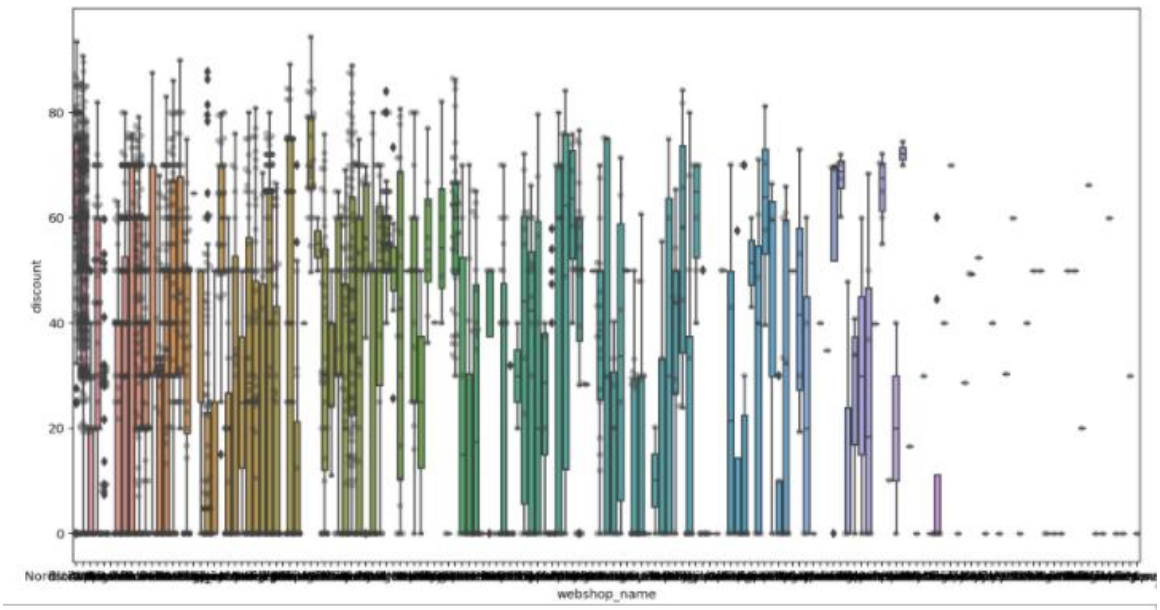
```
Dataframe length before filtering: 428862
Dataframe length  after filtering: 428600
Total filtered rows: 262
```

```
df = df[df.days_alive.dt.days >= 0]
df['days_alive_int'] = df.days_alive.dt.days.astype('int')
```

Se prosigue analizando la variable objetivo descuento (en su formato continuo) respecto a las variables de interés. Como se observa la variable *nombre de la tienda web* presenta una alta dispersión y categorías no representadas. Se determina qué porcentaje de esta variable debe de retenerse sin perder información.

webshop_name

```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='webshop_name', y='discount',data=dfx);
seaborn.swarmplot(x='webshop_name', y='discount',data=dfx, color=".25", alpha=0.4);
```



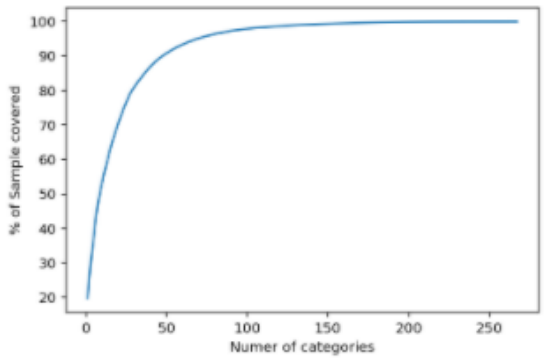
```
x = df[['webshop_name', 'discount']].groupby('webshop_name').count().reset_index()
x = x.sort_values(by='discount', ascending=False)
x.head()
```

	webshop_name	discount
26	Asos	74898
165	Net-A-Porter	21671
103	House of Fraser	18819
170	Nordstrom	15886
171	Nordstrom Rack	15366

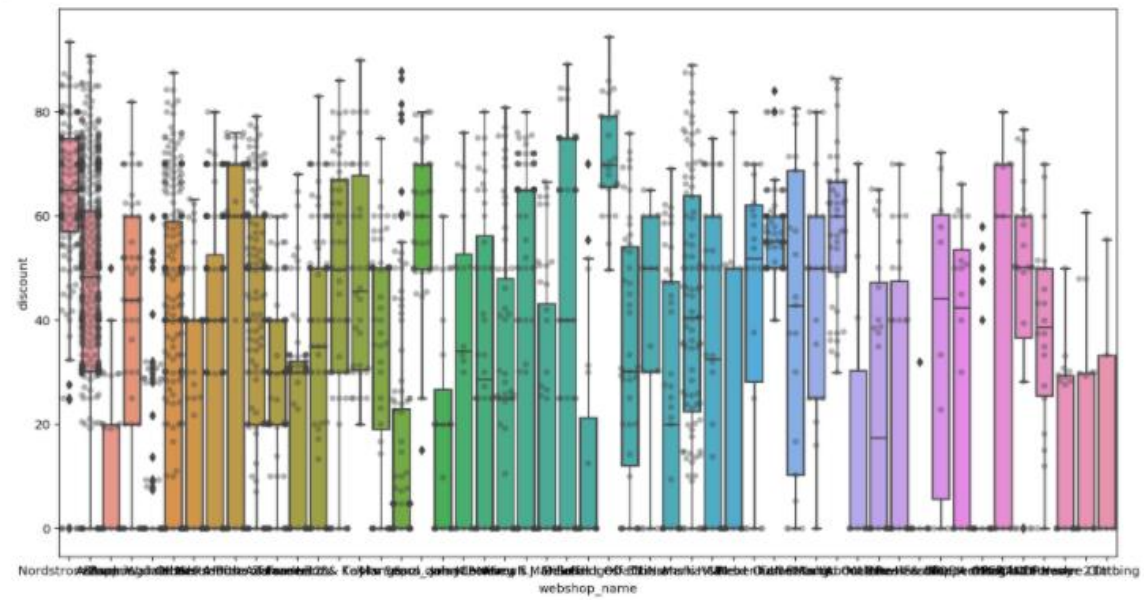
```
v = list()

for i in numpy.arange(len(x),0,-1):
    val = 100 * x.head(i).discount.sum()/x.discount.sum()
    v.append(val)

seaborn.lineplot(numpy.arange(len(x),0,-1), v)
plt.xlabel('Numer of categories')
plt.ylabel('% of Sample covered')
```

```
Text(0, 0.5, '% of Sample covered')
```


```
Text(0, 0.5, '% of Sample covered')
```

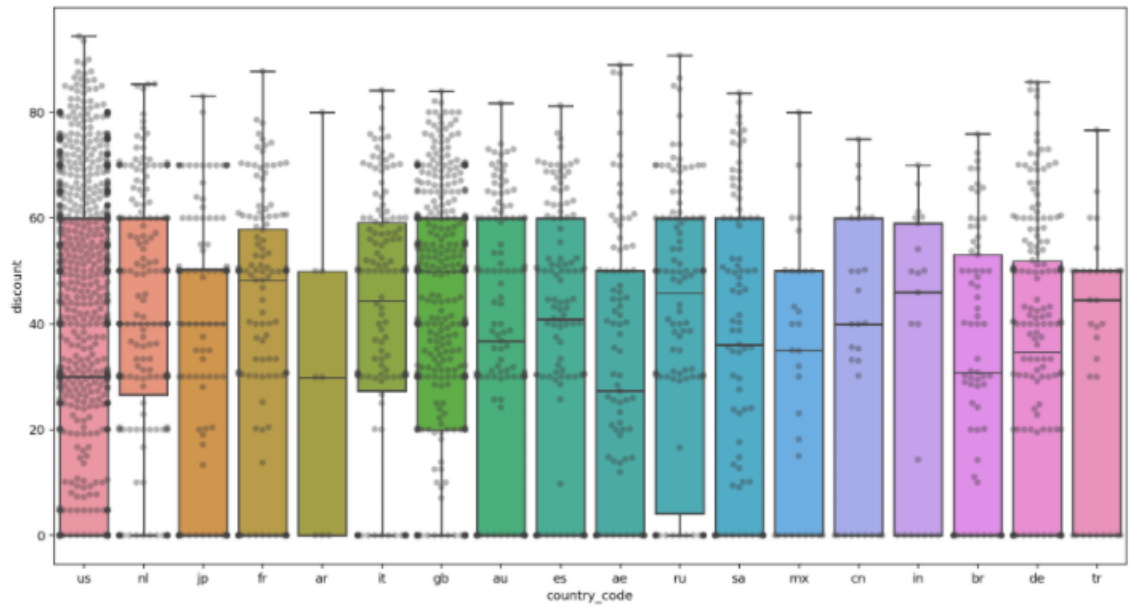



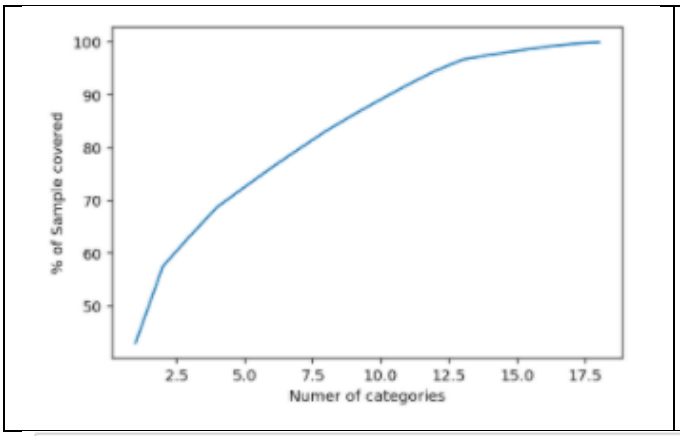
Se identificó que reteniendo 50 categorías de webshop_name se logra representar la muestra en un 90%. A modo informativo entre dichas 50 webs de retailers se encuentran: Asos, Net-A-Porter, Nordstrom, Walmart, Sears, Zalando, Saks, H&M, Macy's, Zozo, Mango, Mango Outlet, Zappos.



Seguidamente se procedió de la misma forma con las variables código de país y marca. Se observó que con 12 países se lograba representar el 90% de la muestra y con 900 marcas.

```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='country_code', y='discount',data=dfx);
seaborn.swarmplot(x='country_code', y='discount',data=dfx, color=".25", alpha=0.4);
```






```
x = df[['country_code', 'discount']].groupby('country_code').count().reset_index()
x = x.sort_values(by='discount', ascending=False)
x.head()
```

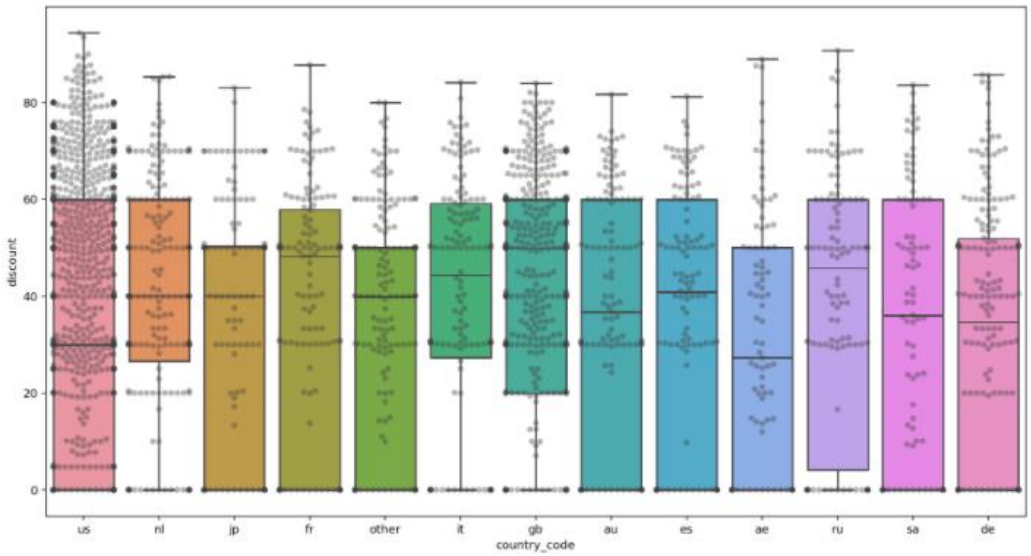
	country_code	discount
17	us	163599
8	gb	55261
5	de	21964
13	nl	20879
2	au	14276

```
v = list()

for i in numpy.arange(len(x),0,-1):
    val = 100 * x.head(i).discount.sum()/x.discount.sum()
    v.append(val)

seaborn.lineplot(numpy.arange(len(x),0,-1), v)
plt.xlabel('Numer of categories')
plt.ylabel('% of Sample covered')
```

```
Text(0, 0.5, '% of Sample covered')
```

```
validcats = x.head(12).country_code.values
df.loc[~df.country_code.isin(validcats),'country_code'] = 'other'
```

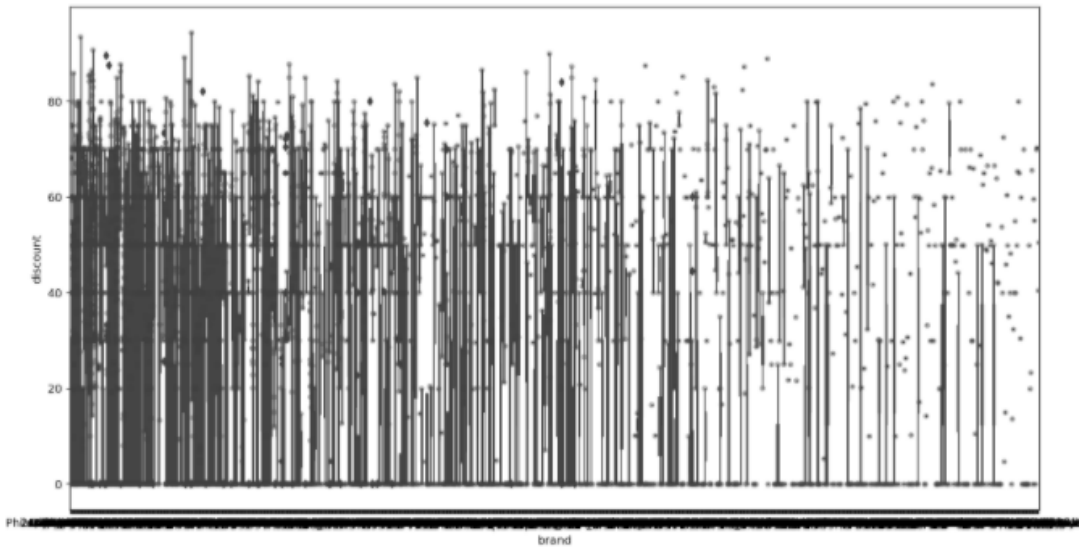
```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='country_code', y='discount',data=dfx);
seaborn.swarmplot(x='country_code', y='discount',data=dfx, color=".25", alpha=0.4);
```



Como se observa en el gráfico anterior los 12 países retenidos son Estados Unidos, Holanda, Japón, Francia, Italia, Gran Bretaña, Australia, España, Alemania, Arabia Saudí, Emiratos y Rusia.

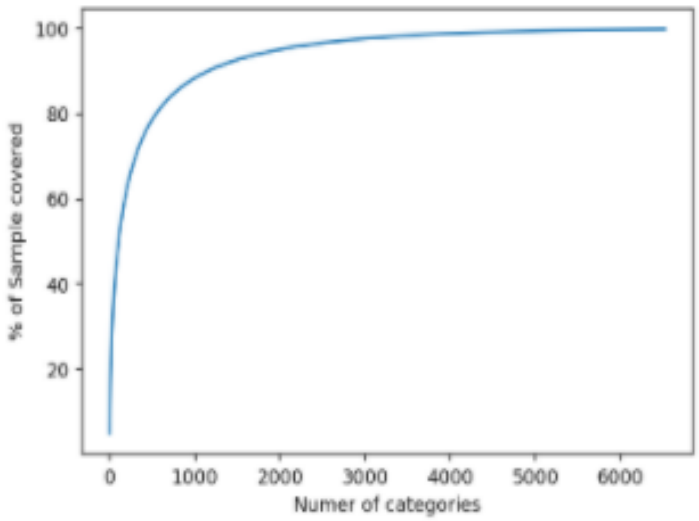
Se realizó el mismo análisis para la variable Marca observándose gran dispersión y como se ha señalado previamente se retuvieron 900 marcas que representan el 90% de la información.

```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='brand', y='discount',data=dfx);
seaborn.swarmplot(x='brand', y='discount',data=dfx, color=".25", alpha=0.4);
```




```
x = df[['brand', 'discount']].groupby('brand').count().
x = x.sort_values(by='discount', ascending=False)
x.head()
```

	brand	discount
534	Asos Private Labels	18557
3658	Mango	11849
3384	Little Mistress	7923
163	Adrianna Papell	6208
2062	Forever21	6111



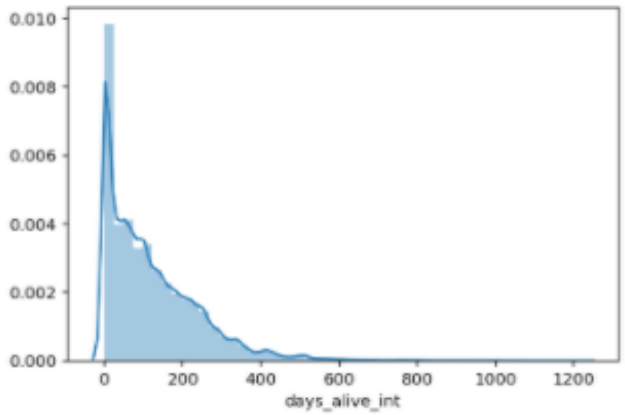

```
validcats = x.head(900).brand.values
df.loc[~df.brand.isin(validcats),'brand'] = 'other'
```

Para la variable días vivos con claro comportamiento exponencial se determinó que se iban a retener aquellos casos de vestidos con una duración de vida de 365 días al año representándose el 90% de la muestra

days_alive_int

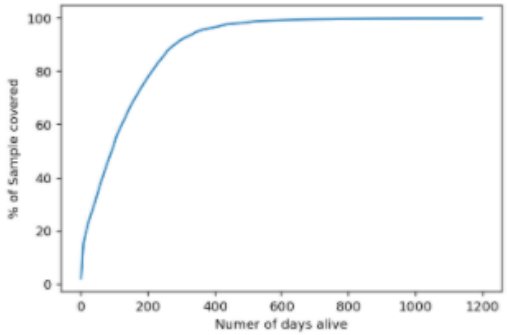
```
seaborn.distplot(df.days_alive_int)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x124360e48>
```




```
N = len(df)
v = list()
for i in numpy.arange(0,1200,7):
    n = len(df[df.days_alive_int <=i])
    #print('Sample Percentage with days_alive_int < {1} $ -> {0:.2f}-{2}'.format(100*n/N, i,n))

    v.append(100*n/N)


seaborn.lineplot(numpy.arange(0,1200,7), v)
plt.xlabel('Numer of days alive')
plt.ylabel('% of Sample covered')
```

```
Text(0, 0.5, '% of Sample covered')
```

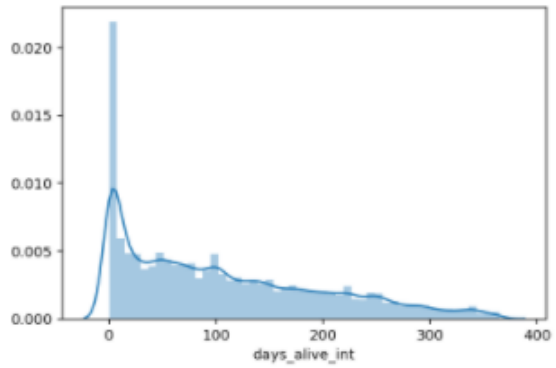


Esta operación permite comprender mejor la información de esta variable como se puede observar en la distribución de la misma:

```
df = df[df.days_alive_int <= 365]
```

```
seaborn.distplot(df.days_alive_int)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x12486c748>
```

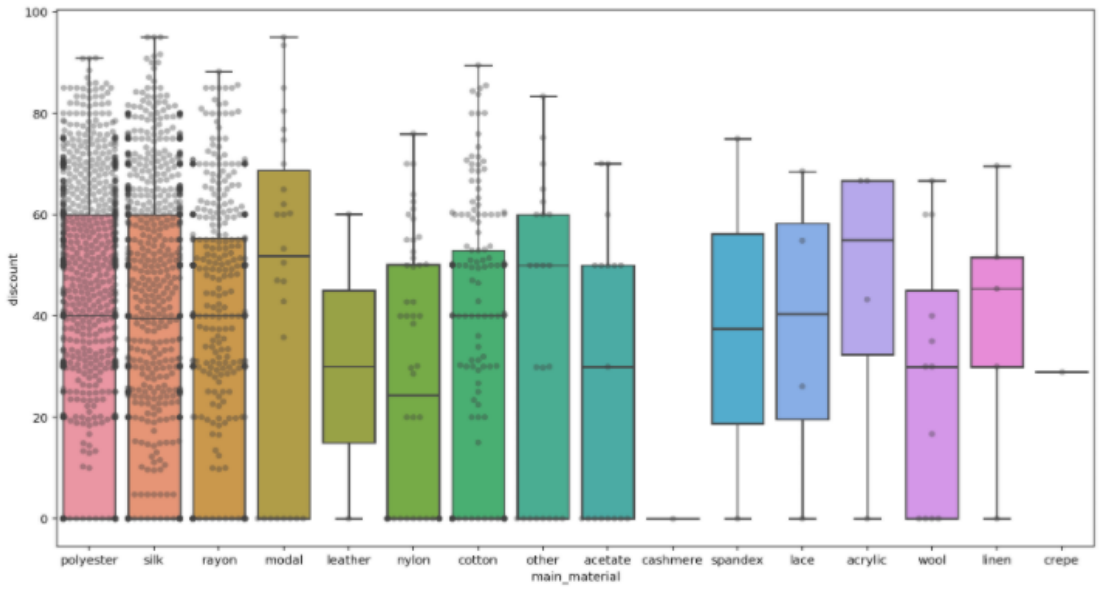


Debido a que la variable material también contiene varias clases, se obtuvo la relación con la variable objetivo y se examinó primero la distribución bivariada. Poliéster, seda, rayón son materiales de alta frecuencia.

```
material_columns = [c for c in df.columns if 'material_' in c]

def remove_material(s):
    return s.split('_')[-1]

df['main_material'] = df[material_columns].idxmax(axis=1).apply(remove_material)
```

```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='main_material', y='discount',data=dfx);
seaborn.swarmplot(x='main_material', y='discount',data=dfx, color=".25", alpha=0.4)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1242a3160>
```



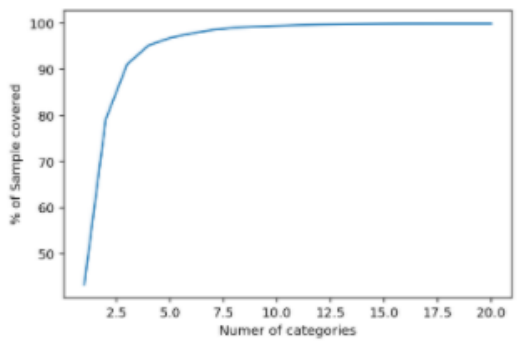
De forma similar a lo descrito con las anteriores variables, se analizó qué número de tipos de material a retener sin sacrificar la representatividad de la muestra, con 9 tipos de materiales se consigue un 90% de representación muestral:

```
x = df[['main_material', 'discount']].groupby('main_material').count().reset_index()
x = x.sort_values(by='discount', ascending=False)
x.head()
```

	main_material	discount
13	polyester	157935
16	silk	130663
14	rayon	43488
3	cotton	14978
11	nylon	5878

```
v = list()

for i in numpy.arange(len(x),0,-1):
    val = 100 * x.head(i).discount.sum()/x.discount.sum()
    v.append(val)

seaborn.lineplot(numpy.arange(len(x),0,-1), v)
plt.xlabel('Numer of categories')
plt.ylabel('% of Sample covered')
```

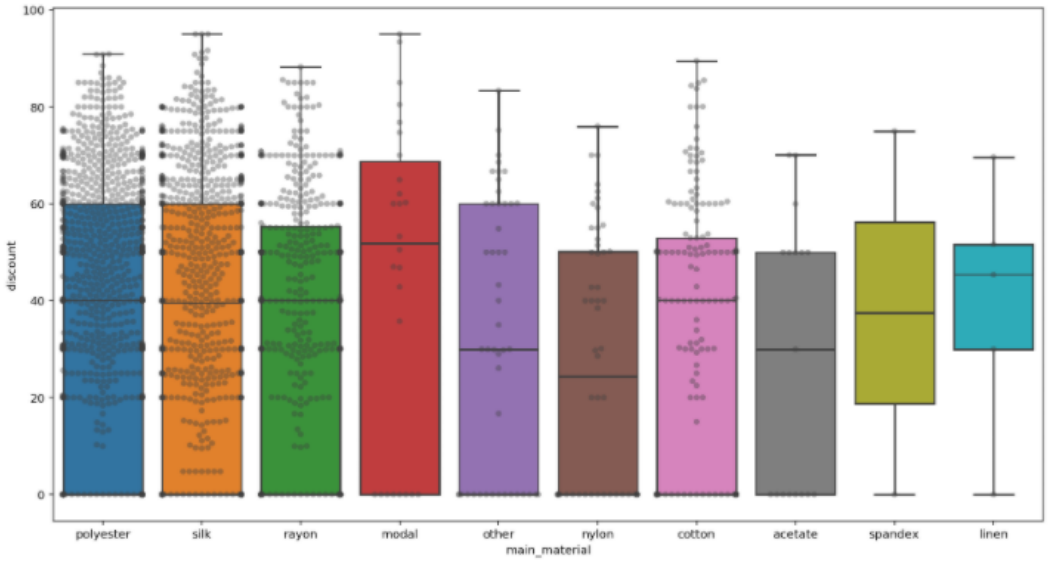
```
Text(0, 0.5, '% of Sample covered')
```




```
validcats = x.head(10).main_material.values
df.loc[~df.main_material.isin(validcats),'main_material'] = 'other'
```

```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='main_material', y='discount',data=dfx);
seaborn.swarmplot(x='main_material', y='discount',data=dfx, color=".25", alpha=0.4)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1276cf5f8>
```



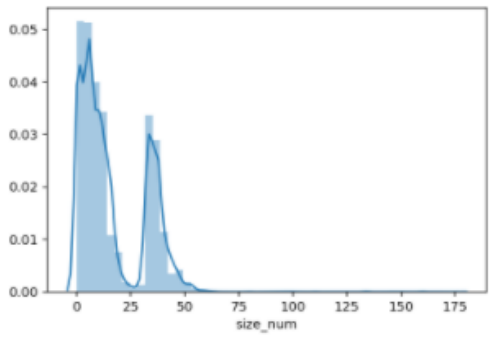
Los materiales retenidos fueron poliéster, seda, rayón, nylon, algodón, acetato, spandex, lino y modal (fibra de rayón modificada).

Respecto a la talla se identificó una distribución bimodal y también se examinó el número de clases de tallas a retener sin perder representación eliminando los valores vacíos.

size_num

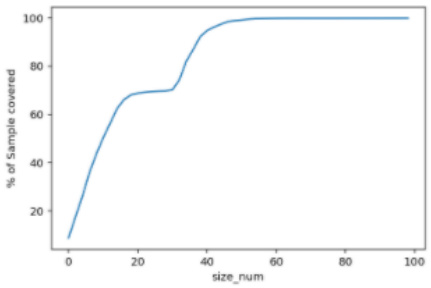
```
seaborn.distplot(df[~df.size_num.isnull()].size_num)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x122c38828>
```




```
N = len(df[~df.size_num.isnull()])
v = list()
for i in numpy.arange(0,100,2):
    n = len(df[df.size_num <=i])
    v.append(100*n/N)


seaborn.lineplot(numpy.arange(0,100,2), v)
plt.xlabel('size_num')
plt.ylabel('% of Sample covered')
```

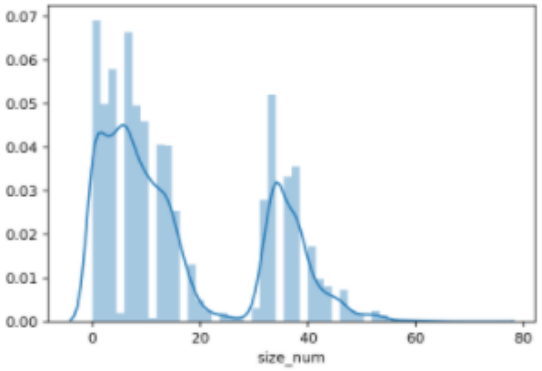
```
Text(0, 0.5, '% of Sample covered')
```




```
df.loc[df.size_num >= 75,'size_num'] = numpy.nan
```

```
seaborn.distplot(df[~df.size_num.isnull()].size_num)
```

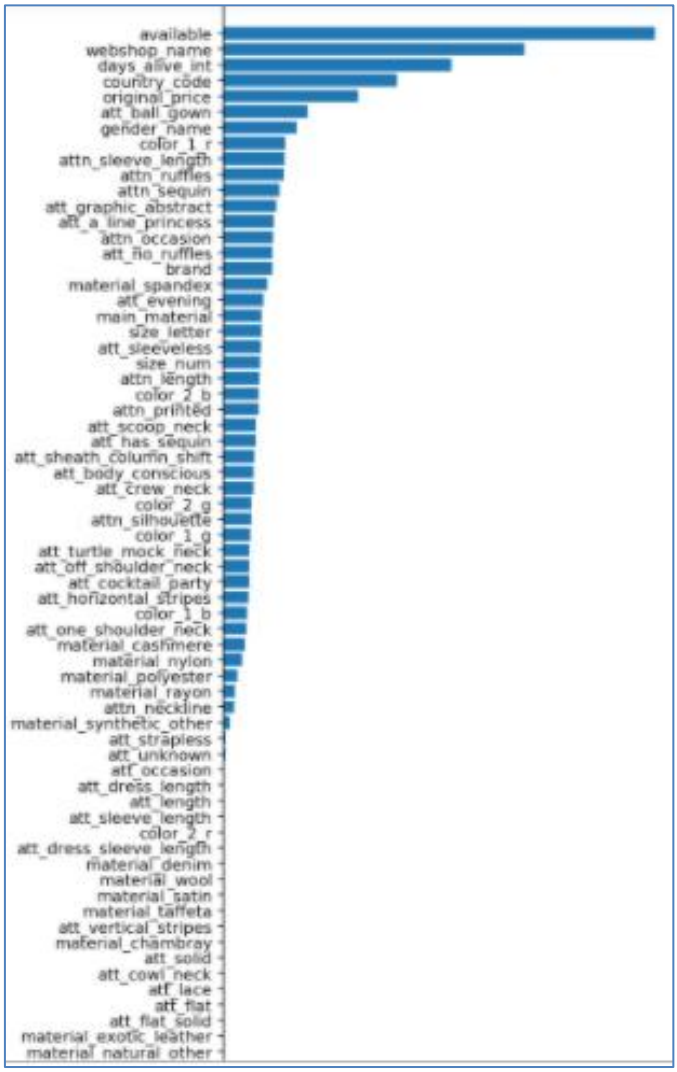
```
<matplotlib.axes._subplots.AxesSubplot at 0x122417860>
```



Más detalle puede consultarse en el Anexo 8.

Antes de la preparación final de los datos se llevó a cabo un estudio de la importancia de las más de 60 variables que resultan del manipulado descrito del conjunto de datos original como muestra el siguiente gráfico, Por último se ha realizado un estudio de selección de variables para determinar la importancia de cada una de ellas a la hora de predecir el precio. A nivel exploratorio resultan potenciales candidatas para incluir en el modelo: disponible, web o retailer, días vivos, país, precio original, género, color, vestido de fiesta (*ball gown*) así como ciertos atributos como longitud de la manga, tener volantes (*ruffles*), llevar lentejuelas (*sequin*), estampado abstracto o vestido corte princesa. Nótese que es una preliminar exploración sin que signifique que estas variables sean las definitivas seleccionadas en el modelo final. En la fase siguiente cuando se prepara el denominado “pipeline” se procede a la selección de variables.

Ilustración 14. Exploración de la importancia de las variables



Fuente: elaboración propia a partir de Python

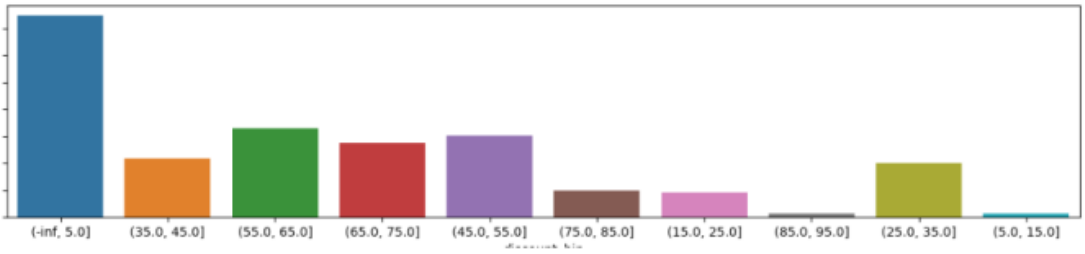
6.4. Preparación final de los datos para el pipeline

Como se detectaron valores atípicos en el atributo *original-price* (prendas a más de 1 millón de dólares o valores negativos) se realizó un filtrado por percentil quedándose con los casos hasta el percentil 95. También prendas con 365 días de vida:

```
def data_selector(df,
                  original_price_percentile=5,
                  countries = 'all',
                  max_days_alive = 365,
                  max_data_size = 75):

    # Percentile for original_price
    pv = numpy.percentile(df.original_price, [original_price_percentile, 100 - original_price_percentile])
    print('Original Price: [{0:.2f},{1:.2f}] - Percentile [{2}]'.format(pv[0],pv[1], original_price_percentile))

    # Filter countries
    if countries == 'all':
        pass
    else:
        df = df.loc[df.country_code.isin(countries)]

    df = df.loc[df.days_alive <= max_days_alive]

    df = df.loc[df.size_num <= max_data_size]

    df = df.loc[df.discount >= 0]

    return df

df_complete_filtered = data_selector(df_complete)

Original Price: [29.99,4210.00] - Percentile [5]
```

Por otro lado, un inconveniente que tiene la modelización es que la variable objetivo creada en categorías de descuento, *discount_bin,* presenta categorías no balanceadas como se muestra a continuación:

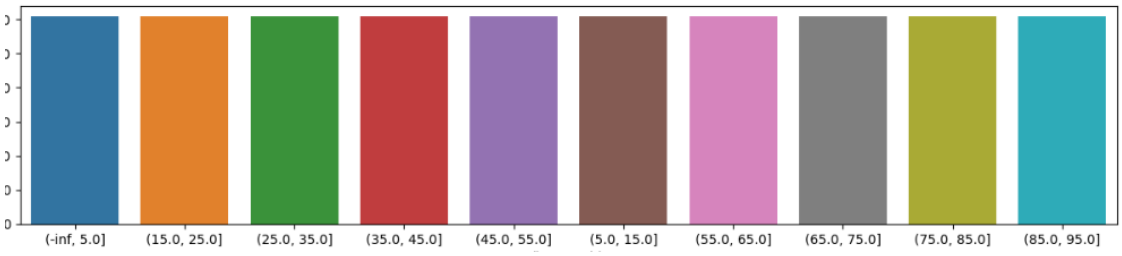
18%	12.8%	15.63%	15.3%	15.63%	4.34%	1.77%	0.47%	14.43%	0.35%



Este problema es ampliamente debatido en la comunidad de data science al ocurrir con frecuencia en conjuntos de datos relacionados con seguros, fraudes fiscales o ataques informáticos. Ello puede conducir a la denominada "trampa de medición"[14] dando lugar a malinterpretaciones de las medidas como el *accuracy_score*. Como posibles medidas se hallan las técnicas de remuestreo denominadas *random under-sampling* y *random over-sampling*. Estas técnicas consisten en eliminar muestras de las clases mayoritarias (*under-sampling*) y/o añadir más ejemplos de la clase minoritaria (*over-sampling*) como se muestra en el anexo 9. Al conjunto de datos de trabajo se ha aplicado en la preparación de los datos finales un remuestreo tipo under-sampling mediante el uso de la librería *imblearn*[15] de tal forma que se logra equilibrar las clases:

14 Véase el siguiente kernel explicativo https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets.

15 Para profundizar véase la librería imbalanced-learn en http://contrib.scikit-learn.org/imbalanced-learn/stable/.



No obstante es preciso señalar que este proceso conlleva a una limitación importante en la modelización ya que eliminar casos aleatorios de las clases más presentes provoca una pérdida de información importante disminuyendo la muestra considerablemente.[16]

Para entrenar el clasificador se ha aplicado el método *cross-validation* o validación cruzada. Primero se ha definido el tipo de *cross-validation* con el número de pliegue, 10 y la cantidad de datos que se usan en el test (35%) en este caso. El método *StratifiedKFold* realiza un muestreo estratificado que genera pliegues que contienen un ratio representativo de cada clase. En cada iteración el código crea un clon del clasificador, entrena el clon en los pliegues de entrenamiento y hace predicciones en el *pliege test*. Entonces cuenta el número de predicciones correctas y arroja el ratio.

Una vez obtenido el mejor clasificador el conjunto de datos este será sometido a una reducción de la dimensionalidad a través de PCA (*Principal Component Analysis*) para comprobar cómo afecta al algoritmo seleccionado. Para ello se va a definir una función cuyos parámetros serán la lista de clasificadores, las estructuras de datos obtenidas por *cross-validation* para llevar a cabo el entrenamiento y el parámetro n indicando cuántas componentes principales se quiere retener. Se vuelve a entrenar el mejor algoritmo ajustando tanto el conjunto de entrenamiento como el de test al número de componentes principales al que queremos reducir.

Los algoritmos entrenados en base al problema de investigación que se programan son: para comenzar el algoritmo *Stochastic Gradient Descent* (SGD) o Descenso Gradiente Estocástico. Este clasificador tiene la ventaja de ser capaz de manejar gran cantidad de datos de forma eficiente. En parte porque el algoritmo SGD trata los casos de entrenamiento de forma independiente de una vez. El nombre estocástico se debe a que el algoritmo confía en la aleatoriedad durante el entrenamiento, gracias a la utilización en el algoritmo del parámetro random_state se puede obtener resultados que son reproducibles. Seguidamente se generan los siguientes algorimos: KNN, SVM, Regresión Logística así como los ensambladores XGBoost y Random Forest.

[16] Entre la depuración y el preparado final del conjunto de datos se han eliminado 32% de los datos

CAPITULO VII. ENTRENAMIENTO Y EVALUACION DE LOS ALGORITMOS

En este capítulo se muestran las pruebas realizadas con los algoritmos citados previamente y su evaluación. Para llevar a cabo el proceso se prepara previamente un *pipeline* de los datos (véase anexo 11). A continuación se muestran para los algoritmos los hiperparámetros empleados así como las métricas y matrices de confusión.

7.1. Evaluación de Stochastic Gradiente Descent (SGD)

Primero se entrena el algoritmo SGD que viene a resolver una regresión logística gradiente estocástico descendente. Si bien se dice que no funciona mejor es útil como punto de partida cuando se tienen grandes cantidades de datos.[17] También se puede señalar que el entrenamiento y test se realiza en menos minutos. El parámetro *random_state* se ha fijado a 42 por ser el que más aparece en los documentos y ejemplos. No existe ninguna explicación sobre porqué la mayoría de los analistas de datos usan ese valor. Una de las posibles explicaciones sobre ello y a título anecdótico se puede encontrar aquí: https://en.wikipedia.org/wiki/The_Hitchhiker%27s_Guide_to_the_Galaxy.

```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train, y_train)

/local_disk0/pythonVirtualEnvDirs/virtualEnv-3c65f1d3-b577-4964-bcc9-f8e0c5e6d527/lib/python3
iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, defau
  FutureWarning)
Out[10]:
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=42, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

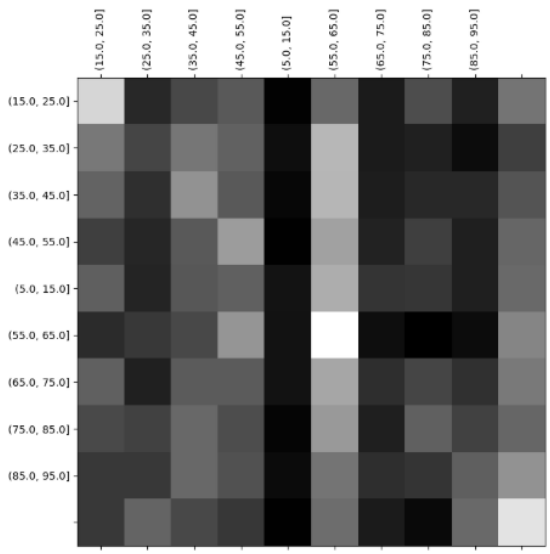
La salida muestra el mejor modelo SGD cuyos hiperparámetros son:

- **Alpha:** 0.0001 es la constante que multiplica el término de regularización.
- **Average:** false significa que el algoritmo no está calculando los pesos ponderados de SGD y almacenando el resultado en el atributo coeficiente. Por ejemplo un valor de 10 indica que el algoritmo empezará a ponderar después de examinar 10 muestras.
- **Class_weight:** none significa que todas las clases se suponen que tiene un peso 1.

[17] Puede profundizarse sobre SGD, entre otras múltiples fuentes, éstas por ser las consultadas en el presente trabajo https://scikit-learn.org/stable/modules/sgd.html y https://towardsdatascience.com/how-to-make-sgd-classifier-perform-as-well-as-logistic-regression-using-parfit-cc10bca2d3c4

- **Epsilon:** 0.1 es el épsilon en la función de pérdida. Este parámetro no es utilizado dado que la función de pérdida es 'hinge'.
- **Eta0:0.0** se refiere a la tasa inicial de aprendizaje para la constante, 0.0 indica que no es utilizado.
- **Fit_intercept:** true indica que la constante ha de ser estimada. Si fuese false el algoritmo asume que los datos ya están centrados.
- **L1_ratio:** 0.15 indica la selección del parámetro elastic net mixing, el valor 0.15
- muestra una regresión ridge imponiendo una penalidad al efecto de las variables sin reducir el número de variables.
- **Loss:** hinge indica que el modelo que se ajusta se controla con el parámetro loss que indica la función de pérdida que en este caso ajusta mediante un Support Vector Machine lineal.
- **Power_t:** 0.5 es el exponente del inverse scaling learning rate.
- **Suffle**: true los datos de entrenamiento son barajados después de cada ronda.
- **Tol:** none indica el criterio de detención. La iteraciones parará cuando loss>best_lost-tol para 5 iteraciones (n_iter_no_change=5)..
- **Validation_fraction:** 0.1 indica el porcentaje de datos de entrenamiento que se apartan como conjunto de validación para el early stopping. Dado que no se utiliza early stopping este porcentaje tampoco se aplica.
- **Verbose:** 0 indica que no se muestra el progreso del aprendizaje automático para no colapsar de salidas.
- **Warm_start:** false indica que el algoritmo no reutiliza la solución de la llamada previa para fijarla como inicialización.

Para evaluar el algoritmo SGD se obtiene la matriz de confusión concluyendo en unos resultados pobres como muestran las métricas de precisión, recall, f1-score y support todas ellas con valores por debajo de 0.20.



	precision	recall	f1-score
(-inf, 5.0]	0.20	0.23	0.22
(15.0, 25.0]	0.12	0.10	0.11
(25.0, 35.0]	0.14	0.17	0.15
(35.0, 45.0]	0.15	0.18	0.16
(45.0, 55.0]	0.13	0.04	0.07
(5.0, 15.0]	0.15	0.28	0.20
(55.0, 65.0]	0.12	0.07	0.09
(65.0, 75.0]	0.16	0.12	0.14
(75.0, 85.0]	0.17	0.12	0.14
(85.0, 95.0]	0.17	0.25	0.20
micro avg	0.16	0.16	0.16
macro avg	0.15	0.16	0.15
weighted avg	0.15	0.16	0.15

- En relación a la precisión los valores se alejan considerablemente de 1 sin superar el 20%.
- La medida recall, la tasa de verdaderos positivos, arroja valores similares siendo el mayor valor, 0.28 en el intervalo de descuento entre 5 y 15.
- Dado que la medida f1-score realiza una media armónica entre precisión y recall los resultados son igual de pobres.

7.2. Evaluación de la Regresión logística

Entrenado el algoritmo SGD se procede a utilizar el clasificador regresión logística[18], un método tradicional que suele tener buenos resultados. El modelo mejor es el que aparece en la siguiente salida:

```
from sklearn.linear_model import LogisticRegression

softmax_reg = LogisticRegression(multi_class="multinomial",solver="lbfgs")
softmax_reg.fit(X_train, y_train)

/local_disk0/pythonVirtualEnvDirs/virtualEnv-3c65f1d3-b577-4964-bcc9-f8e0c5e6d527/lib
  "of iterations.", ConvergenceWarning)
Out[20]:
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='multinomial',
          n_jobs=None, penalty='l2', random_state=None, solver='lbfgs',
          tol=0.0001, verbose=0, warm_start=False)
```

Respecto a los hiperparámetros del algoritmo se especifica:

- **Multi_class:** la pérdida minimizada es la pérdida multinomial en toda la distribución de probabilidad frente a OvR.
- **Solver:** En este caso se ha optado por el método lbfgs.

El resultado de las métricas es ligeramente mejor en algunas clases y a nivel global pero sigue sin ser satisfactorio.

[18] Para consultas sobre el clasificador en scikit y los hiperparámetros consultar https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

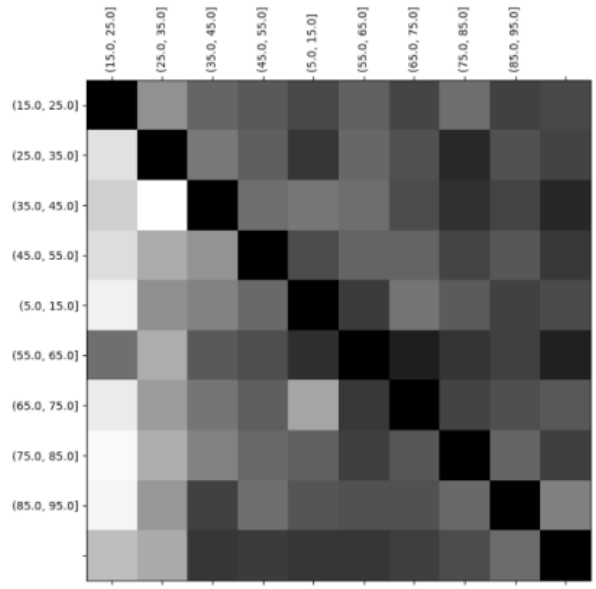
	precision	recall	f1-score
(-inf, 5.0]	0.23	0.33	0.27
(15.0, 25.0]	0.20	0.18	0.19
(25.0, 35.0]	0.21	0.20	0.20
(35.0, 45.0]	0.17	0.10	0.13
(45.0, 55.0]	0.17	0.12	0.14
(5.0, 15.0]	0.25	0.46	0.32
(55.0, 65.0]	0.14	0.09	0.11
(65.0, 75.0]	0.15	0.11	0.13
(75.0, 85.0]	0.13	0.11	0.12
(85.0, 95.0]	0.22	0.32	0.26
micro avg	0.20	0.20	0.20
macro avg	0.19	0.20	0.19
weighted avg	0.19	0.20	0.19

7.3. Evaluación de KNN, K vecinos más cercanos

Vistos los pobres resultados de los clasificadores anteriores se entrena el algoritmo KNN[19] en lugar del RadiusNN. Estos se conocen en ML como “no generalizables” porque lo que hacen simplemente es recordar todos sus datos de entrenamiento. Aunque son muy simples es uno de los algoritmos más usados y por ser no paramétrico puede ser útil en situaciones irregulares como ésta. La búsqueda de los vecinos por el algoritmo se controla por el hiperparámetro algorithm que en este caso es auto, quiere decir que el clasificador espera determinar el mejor resultado desde los datos de entrenamiento. Este es el mejor KNN pero se observa en sus métricas y matriz que los resultados del clasificador tampoco son óptimos a pesar de mejorar en una de las clases sustancialmente:

```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)

Out[17]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

[19] https://scikit-learn.org/stable/modules/neighbors.html




```
print(classification_report(y_test, knn_clf.pre

                precision    recall  f1-score   s

  (-inf, 5.0]       0.16      0.31      0.21
(15.0, 25.0]        0.18      0.28      0.22
(25.0, 35.0]        0.15      0.14      0.14
(35.0, 45.0]        0.18      0.14      0.16
(45.0, 55.0]        0.19      0.15      0.16
 (5.0, 15.0]        0.42      0.43      0.43
(55.0, 65.0]        0.16      0.12      0.14
(65.0, 75.0]        0.16      0.11      0.13
(75.0, 85.0]        0.17      0.12      0.14
(85.0, 95.0]        0.37      0.31      0.34

    micro avg       0.21      0.21      0.21
    macro avg       0.21      0.21      0.21
weighted avg        0.21      0.21      0.21
```

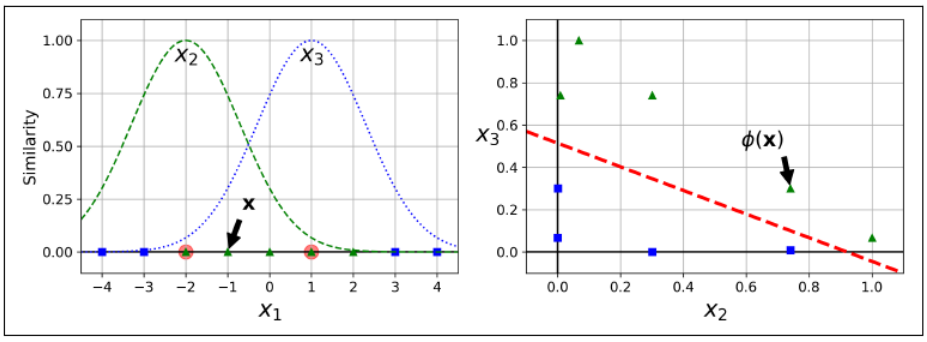
7.4. Evaluación Support Vector Machine

A continuación se entrena un nuevo algoritmo, el SVM[20], se ha elegido en este caso el Gaussian RBF kernel[21] que utiliza una función de similitud que mide cuanto se parece cada caso a un punto de referencia. Esta función es Gaussian Radial Basis Function (RBF) con la siguiente fórmula:

$$\phi_\gamma(\mathbf{x}, \ell) = \exp\left(-\gamma \| \mathbf{x} - \ell \|^2\right)$$

Esta función de similitud tiene una forma de campana que varía de 0 (muy alejado del punto de referencia) a 1 (en el punto de referencia). El hiperparámetro gamma es un regularizador.

Ilustración 14. Gaussian Radial Basis Function-RBF



Fuente: O'Really

[20] https://scikit-learn.org/stable/modules/svm.html
[21] https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RBF.html y https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

```
from sklearn.svm import SVC
rbf_kernel_svm_clf = SVC(kernel="rbf",gamma='auto')
rbf_kernel_svm_clf.fit(X_train, y_train)
print(classification_report(y_test, rbf_kernel_svm_clf.predict(X_test), target
```

Como se pude observar las métricas no son satisfactorias y resultan similares a las previas:

```
              precision    recall  f1-score

 (-inf, 5.0]       0.23      0.27      0.25
(15.0, 25.0]       0.23      0.22      0.23
(25.0, 35.0]       0.21      0.24      0.22
(35.0, 45.0]       0.21      0.13      0.16
(45.0, 55.0]       0.19      0.15      0.17
 (5.0, 15.0]       0.30      0.52      0.38
(55.0, 65.0]       0.17      0.12      0.14
(65.0, 75.0]       0.22      0.15      0.18
(75.0, 85.0]       0.18      0.17      0.17
(85.0, 95.0]       0.32      0.38      0.35

   micro avg       0.24      0.24      0.24
   macro avg       0.22      0.24      0.22
weighted avg       0.23      0.24      0.22
```

7.5. Evaluación de Random Forest

Se procede a probar algoritmos de ensamblado con la expectativa de que logar acercarse al objetivo propuesto de forma satisfactoria. Primero se entrena el clasificador Random Forest[22] el metaestimador que ajusta un número de árboles de clasificación a varias submuestras del conjunto de datos y usa promedios para mejorar la capacidad predictiva y controlar el sobreajuste. A continuación se muestra el algoritmo entrenado y sus hiperparámetros:

```
from sklearn.ensemble import RandomForestClassifier

#forest_clf = RandomForestClassifier(random_state=42)
forest_clf = RandomForestClassifier(bootstrap=False, class_weight=None, criterion='gini',
             max_depth=15, max_features='auto', max_leaf_nodes=None,
             min_impurity_decrease=0.0, min_impurity_split=None,
             min_samples_leaf=12, min_samples_split=9,
             min_weight_fraction_leaf=0.0, n_estimators=2000, n_jobs=-1,
             oob_score=False, random_state=42, verbose=0, warm_start=False)



forest_clf.fit(X_train, y_train)
```

[22] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Y como arrojan las métricas los resultados mejorar significativamente como se observa en las métricas del conjunto entrenamiento pero no en el conjunto test:

	precision	recall	f1-score
(-inf, 5.0]	0.79	0.58	0.67
(15.0, 25.0]	0.52	0.74	0.61
(25.0, 35.0]	0.56	0.63	0.59
(35.0, 45.0]	0.77	0.55	0.64
(45.0, 55.0]	0.73	0.58	0.65
(5.0, 15.0]	0.57	0.82	0.67
(55.0, 65.0]	0.83	0.53	0.65
(65.0, 75.0]	0.74	0.53	0.62
(75.0, 85.0]	0.66	0.63	0.65
(85.0, 95.0]	0.53	0.77	0.63
micro avg	0.64	0.64	0.64
macro avg	0.67	0.64	0.64
weighted avg	0.67	0.64	0.64

```
print(classification_report(y_test, forest_clf
```

	precision	recall	f1-score
(-inf, 5.0]	0.39	0.26	0.31
(15.0, 25.0]	0.30	0.46	0.37
(25.0, 35.0]	0.22	0.30	0.25
(35.0, 45.0]	0.36	0.21	0.26
(45.0, 55.0]	0.23	0.18	0.20
(5.0, 15.0]	0.42	0.69	0.52
(55.0, 65.0]	0.34	0.16	0.22
(65.0, 75.0]	0.35	0.22	0.27
(75.0, 85.0]	0.31	0.26	0.28
(85.0, 95.0]	0.38	0.59	0.46
micro avg	0.33	0.33	0.33
macro avg	0.33	0.33	0.32
weighted avg	0.33	0.33	0.31

7.6. Evaluación de XGboost

Seguidamente se entrena el XGboost cuyas siglas provienen de Extreme Gradient Boosting[23]. Es uno de los algorimos más de moda en la actualidad y muy utilizado en las competiciones de Kaggel por sus resultados y la velocidad de ejecución. Puede observarse que lo resultados no son satisfactorios ni en datos de entrenamiento ni test.

```
from xgboost import XGBClassifier
xgb_clf = XGBClassifier()

xgb_clf.fit(X_train,y_train)
```

	precision	recall	f1-score
(-inf, 5.0]	0.39	0.26	0.31
(15.0, 25.0]	0.30	0.46	0.37
(25.0, 35.0]	0.22	0.30	0.25
(35.0, 45.0]	0.36	0.21	0.26
(45.0, 55.0]	0.23	0.18	0.20
(5.0, 15.0]	0.42	0.69	0.52
(55.0, 65.0]	0.34	0.16	0.22
(65.0, 75.0]	0.35	0.22	0.27
(75.0, 85.0]	0.31	0.26	0.28
(85.0, 95.0]	0.38	0.59	0.46
micro avg	0.33	0.33	0.33
macro avg	0.33	0.33	0.32
weighted avg	0.33	0.33	0.31

```
print(classification_report(y_test, xgb_clf.pr
```

	precision	recall	f1-score
(-inf, 5.0]	0.34	0.34	0.34
(15.0, 25.0]	0.25	0.43	0.31
(25.0, 35.0]	0.20	0.24	0.22
(35.0, 45.0]	0.37	0.17	0.24
(45.0, 55.0]	0.19	0.16	0.17
(5.0, 15.0]	0.40	0.62	0.49
(55.0, 65.0]	0.32	0.17	0.22
(65.0, 75.0]	0.34	0.21	0.26
(75.0, 85.0]	0.27	0.21	0.24
(85.0, 95.0]	0.44	0.55	0.49
micro avg	0.31	0.31	0.31
macro avg	0.31	0.31	0.30
weighted avg	0.31	0.31	0.30

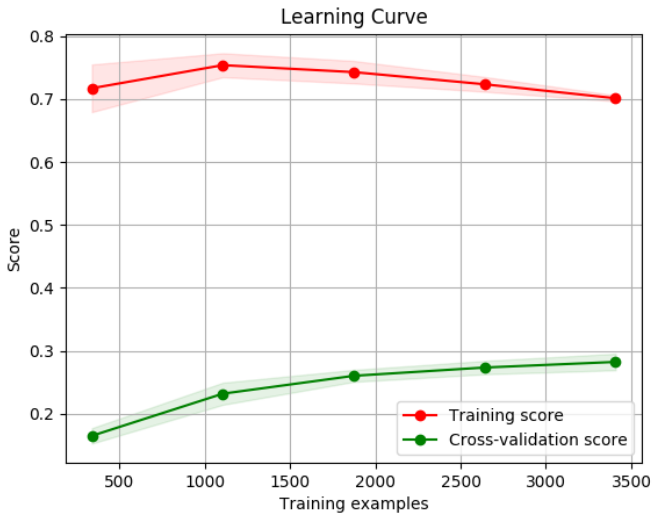
[23] Una fuente recomendada es https://github.com/dmlc/xgboost y en https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn

7.7. Random forest: análisis e importancia de las variables.

Tras realizar los entrenamientos de los clasificadores descritos previamente, se obtiene como el mejor en rendimiento el clasificador random forest. Se llevan a cabo tres tareas más y seguidamente se procede a obtener las reglas de clasificación.

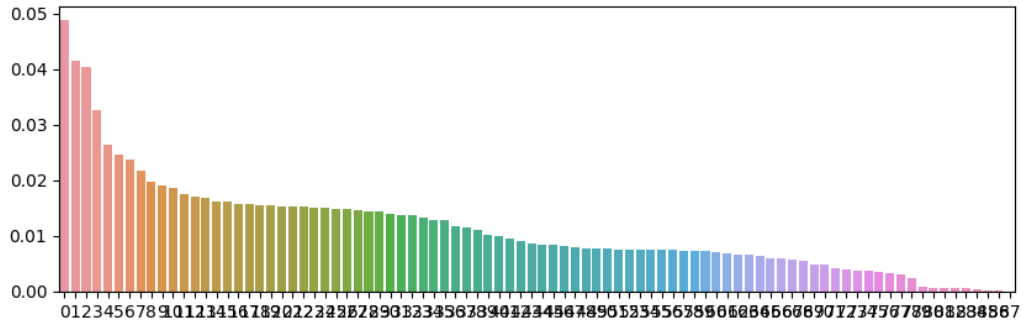
La primera es dado que en el conjunto test los resultados siguen siendo poco prometedores es observar la **curva de aprendizaje** para ver como se mejorarían las métricas aumentando el número de muestras, se puede identificar que esto no lo mejoraría:

```
cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)

fig = plot_learning_curve(forest_clf, 'Learning Curve', X_test, y_test, ylim=None, cv=cv, n_jobs=-1)
display(fig)
```



El segundo paso es probar una **reducción de dimensionalidad con un PCA.** Entrenando el algoritmo con Random Forest y Componentes Principales tampoco mejoran los resultados como se puede observar:

```
n_components=numpy.round(X_new.shape[1]/2).astype('int')
pca = PCA(n_components=n_components)
pca.fit(X_new)

fig, ax = plt.pyplot.subplots(figsize=(10,3))
seaborn.barplot(y=pca.explained_variance_ratio_, x=numpy.arange(0,n_components))
display(plt.pyplot.show())

X_pca = pca.fit_transform(X_new)
```

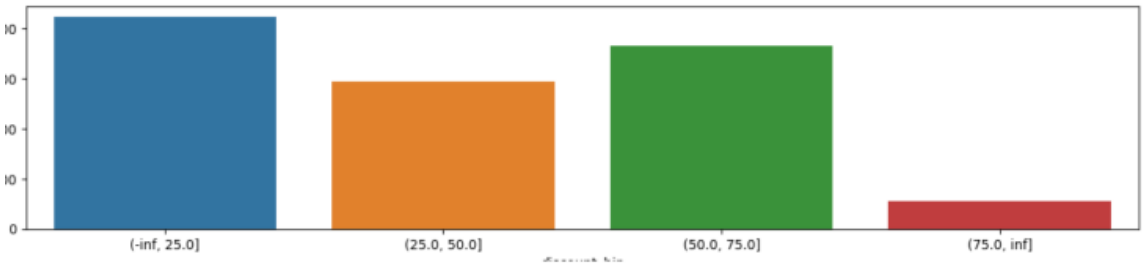



```
print(classification_report(y_test, random_sea
```

	precision	recall	f1-score
(-inf, 5.0]	0.22	0.23	0.23
(15.0, 25.0]	0.28	0.27	0.27
(25.0, 35.0]	0.28	0.29	0.29
(35.0, 45.0]	0.22	0.20	0.21
(45.0, 55.0]	0.17	0.20	0.18
(5.0, 15.0]	0.39	0.57	0.46
(55.0, 65.0]	0.27	0.11	0.16
(65.0, 75.0]	0.26	0.16	0.20
(75.0, 85.0]	0.21	0.19	0.20
(85.0, 95.0]	0.36	0.52	0.43
micro avg	0.28	0.28	0.28
macro avg	0.27	0.28	0.26
weighted avg	0.27	0.28	0.26

En tercer lugar **se decide probar con menos clases**, en lugar de las 10 clases de porcentaje de descuento en precio con la que se ha trabajado, se entrenan random forest para 4 clases y se vuelve a preparar otro pipeline:

```
    else:
        df = df.loc[df.country_code.isin(countries)]

    df = df.loc[df.days_alive <= max_days_alive]

    df = df.loc[df.size_num <= max_data_size]

    df = df.loc[df.discount >= 0]

    ## Create categorical target variable
    '''
    intervals = pandas.IntervalIndex.from_tuples([(-numpy.inf, 5),
                                                  (5, 15),
                                                  (15, 25),
                                                  (25, 35),
                                                  (35, 45),
                                                  (45, 55),
                                                  (55, 65),
                                                  (65, 75),
                                                  (75, 85),
                                                  (85, 95),
                                                  (95, 100)])
    '''

    ## Create categorical target variable
    intervals = pandas.IntervalIndex.from_tuples([(-numpy.inf, 25),
                                                  (25, 50),
                                                  (50, 75),
                                                  (75, numpy.inf)])

    df['discount_bin'] = pandas.cut(df.discount, bins=intervals)

    return df

df_complete_filtered = data_selector(df_complete)
```



Estos son los hiperparámetros del algoritmo:

```
from sklearn.ensemble import RandomForestClassifier

#forest_clf = RandomForestClassifier(random_state=42)
forest_clf = RandomForestClassifier(bootstrap=False, class_weight=None, criterion='gini',
             max_depth=15, max_features='auto', max_leaf_nodes=None,
             min_impurity_decrease=0.0, min_impurity_split=None,
             min_samples_leaf=12, min_samples_split=9,
             min_weight_fraction_leaf=0.0, n_estimators=2000, n_jobs=-1,
             oob_score=False, random_state=42, verbose=0, warm_start=False)



forest_clf.fit(X_train, y_train)

Out[15]:
RandomForestClassifier(bootstrap=False, class_weight=None, criterion='gini',
            max_depth=15, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=12, min_samples_split=9,
            min_weight_fraction_leaf=0.0, n_estimators=2000, n_jobs=-1,
            oob_score=False, random_state=42, verbose=0, warm_start=False)
```

Como se observa el último agrupamiento mejora las métricas en los datos test

	precision	recall	f1-score
(-inf, 25.0]	0.59	0.80	0.68
(25.0, 50.0]	0.67	0.31	0.42
(50.0, 75.0]	0.56	0.65	0.60
(75.0, inf]	0.95	0.01	0.01
micro avg	0.59	0.59	0.59
macro avg	0.69	0.44	0.43
weighted avg	0.62	0.59	0.56

Finalmente, obtenido el algoritmo que mejor clasifica se obtiene el análisis de explicabilidad para poder intepretar el modelo (figura 15). Para ello se ha utilizado la librería SHAP: *Shapley Additive Explanation*.[24] Este concepto se basa en la teoría de juegos para explicar cómo cada una de las variables que entran en juego intervienen en un "juego colaborativo" contribuyen al éxito. De esta forma se pueden comprender las predicciones y cómo afecta cada variable.

```
import joblib
import shap
import numpy
from joblib import dump, load
shap.initjs()
```

```
forest_clf = load('randomForest_final.joblib')
```

```
explainer = shap.TreeExplainer(forest_clf)
```

```
shap_values = explainer.shap_values(X.sample(20))
```

```
shap.summary_plot(shap_values, X)
```

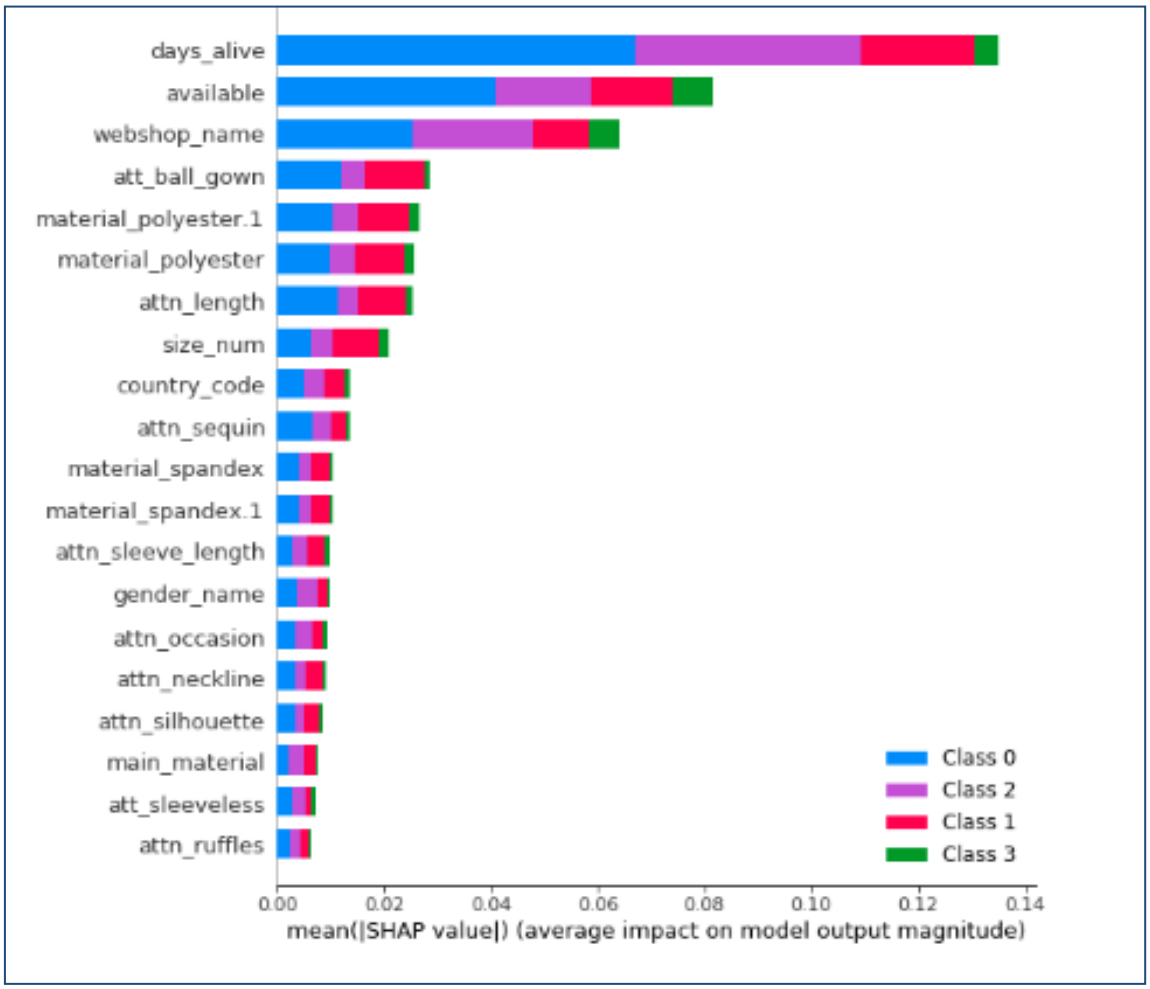
El conjunto de todos los valores SHAP (véase anexo 11) permite obtener la gráfica de la contribución de cada variable. Como se observa 20 características de las prendas entran en el modelo construido para clasificar prendas a partir del intervalo de descuento en precio con los siguientes comentarios:

- Tras haber creado la variable objetivo porcentaje de descuento en precio, el mejor resultado se obtiene con la variable objetivo que consiste en 4 clases, la clase 0, descuentos en precio entre 0 y 25%, la clase 1, descuentos entre 25 y 50% sobre el precio original, la clase 2, descuentos entre el 50 y el 75% y la clase 3 descuentos superiores a 75% sobre el precio origina de la prenda.
- La clase 0 representa el 19.47% de los datos, la clase 1, el 28% de los datos, las clase 2 el 47.2% de los datos y la clase 3, la de los descuentos más agresivos el 4.97% de los datos.
- Casi la mitad de las prendas de vestidos de mujer son vendidas a un precio de descuento entre el 50-75%

[24] Véase https://christophm.github.io/interpretable-ml-book/shap.html

- La variable con mayor importancia para explicar la variable objetivo es el número de días vivos de la prenda con un impacto en media de 14%. Además esta tiene un impacto en media en la predicción de 6% en la clase 0.
- El segundo lugar lo ocupa la variable si la prenda sigue disponible o no impactando en 8.2%
- La tercera variable en importancia de clasificación de descuentos es el retailer (impacto del 6%).
- El cuarto atributo más importante es si el vestido es de fiesta con un porcentaje de impacto del 3% en media.
- Los siguientes atributos en importancia similar son: si el vestido es de fiesta, si el material es poliéster, la longitud del vestido y el número de talla.

Ilustración 15. Importancia de las variables en la clasificación



CAPITULO VIII. CONCLUSIONES, DISCUSION, LIMITACIONES Y LINEAS FUTURAS.

8.1. Conclusiones y discusión

El objetivo que se perseguía en este trabajo era doble. Por un lado, tratar de clasificar el porcentaje de descuento en precio aplicado a prendas de ropa, vestidos, a partir de una serie de atributos. Por otro lado, llevar a cabo distintas técnicas y estrategias de clasificación usando la biblioteca de aprendizaje automático scikit-learn para el lenguaje de programación Python. Se ha utilizado un conjunto de datos de más de 400.000 casos cedidos por la empresa tecnológica StyleSage de Big data que opera en el sector de la moda. El propósito era obtener distintas métricas para evaluar y analizar cada una de las técnicas. De cada técnica y para cada algoritmo empleado se ha realizado una búsqueda exhaustiva para encontrar los hiperparámetros que permitan obtener los mejores resultados sobre los datos y, a continuación, se ha realizado un entrenamiento de cada uno de ellos con dichos hiperparámetros para así poder realizar una clasificación. En el proceso se ha usado una técnica de validación cruzada en la que se obtiene un número de ejemplos para poder entrenar el algoritmo y otro número de ejemplos, denominados validación, para probar cada algoritmo entrenado y obtener métricas. Para procesar los datos, entrenar los algoritmos y obtener las medidas de evaluación y etiquetado se ha desarrollado un programa en el lenguaje Python ilustrado paso por paso a lo largo del documento. Se ha podido mostrar así que el aprendizaje supervisado se compone tres etapas hasta llegar a un modelo aceptado como se muestra en el anexo 13.

Primero se ha completado un preprocesado de los datos para conocer la naturaleza de los datos, su distribución, datos perdidos y atributos que corresponden a la etiqueta de clasificación. Dado que los algoritmos algunos requieren estandarización o normalización así como centralizarlos, se ha analizado cada caso. El segundo paso ha sido escoger y entrenar una serie distinta de algoritmos de clasificación: 1.Búsqueda de la estrategia a usar para realizar la validación cruzada. 2. Optimización de los hiperparámetros para el algoritmo. La biblioteca scikit-learn posibilita una búsqueda exhaustiva en un espacio finito para obtener estos hiperparámetros. 3. Cuantificación de la calidad de las predicciones realizadas por el algoritmo tras el entrenamiento de cada uno de ellos y con el uso de validación cruzada. Finalmente, se ha completado la fase de uso del modelo obtenido: se ha podido realizar la clasificación para nuevos datos que no han sido vistos previamente, datos que no tienen etiqueta y que se quieren clasificarlos usando el algoritmo seleccionado habiendo entrenado previamente.

De todo este proceso de ML se pueden extraer varias conclusiones. La primera es que la etapa de procesado es compleja, requiere gran cantidad de tiempo y es necesaria para tener calidad en los algoritmos. No es un paso mecánico porque requiere analizar y reflexionar bien sobre las estrategias para preparar los datos para el pipeline definitivo que entrena el algoritmo. Muchas veces se tienen datos que aparentemente son muy valiosos y la realidad es que están llenos de ruido y de información que no es útil. Es por ello que se recomienda invertir mucho tiempo en el

preprocesado de datos y asegurarse que se ha completado cualquier detalle. Incluso quizás tras el preprocesado es recomendable volver a reflexionar sobre si el conjunto de datos debe de ser enriquecido o no. Además el preprocesado de datos puede dar indicaciones para confirmar si el objetivo que se ha planteado es acertado o no y si se debe de proseguir.

Por otro lado, se detecta que es determinante la búsqueda de los hiperparámetros para cada uno de los algoritmos. También el tipo de variables si requieren estandarización o no para el algoritmo como es el caso de la regresión logística. De ello se ha concluido que la capacidad del equipo computacional es clave para esta etapa porque permite ahorrar tiempo y evaluar un mayor número de posibilidades mayor para cada algoritmo.

Finalmente, aunque se dice que la técnica de reducción de la dimensión de los datos mediante PCA (Principal Component Analysis) influye en los algoritmos y puede mejorarlos, se ha podido comprobar que en este caso no ha sido así la mejoría. Bien es cierto que solo se ha probado en Random forest dado que se ha concluido que es el mejor algoritmo a usar para el conjunto de datos y se ha logrado el objetivo principal de clasificar como muestran las reglas de clasificación. Cabe señalar que los resultados están por debajo del 80 y 90% que serían niveles satisfactorios.

8.2. Interpretabilidad e implicación empresarial

En relación a los resultados y la interpretación de los mismos así como su implicación empresarial se extraen las siguientes conclusiones:

- El mejor algoritmo resultante, random forest, es el aplicado a la variable objetivo porcentaje de descuento clasificada en 4 categorías (descuentos por debajo de 25%), descuentos entre 25 y 50% del precio original, descuentos de 50 a 75% y una cuarta categoría que engloba prendas con descuento por encima de 75% en precio.
- La categoría con mayor frecuencia es la de los descuentos entre 50 y 75%. Esto es coherente con el negocio de la moda en el que la eliminación de los stocks resulta clave para la rentabilidad del negocio y por la naturaleza de la moda: estacionalidad y rapidez en el cambio de los diseños.
- Del conjunto de variables con el que se partía el modelo integra 20 atributos que contribuyen a la clasificación de las prendas.
- Los dos primeros atributos días vivos y disponibilidad de la prenda hacen referencia al ciclo de vida del producto. El atributo más importante para las predicciones es el número de días vivos. Le sigue la disponibilidad de la prenda.
- El retailer y si el vestido es de fiesta (también ocasión en un nivel inferior) son las siguientes en importancia para explicar las clases de descuento.
- Asimismo aparecen atributos relacionados con el material: poliéster, spandex.
- Junto a ellos se incluyen atributos relacionados con el diseño como la presencia de lentejuelas, volantes o cuello a la caja.

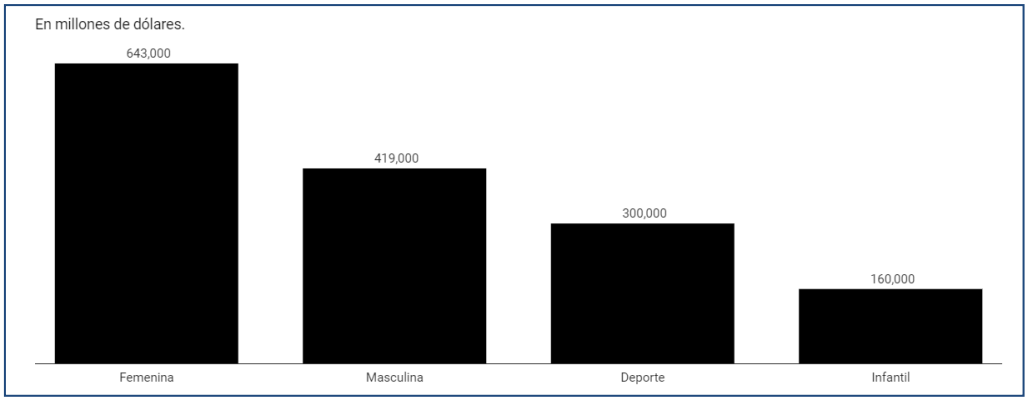
8.3. Limitaciones y futuras líneas

Este trabajo, como cualquier otro no, está exento de limitaciones. La primera limitación que hay que indicar es que solo se ha analizado una categoría de prendas: vestidos. Se han excluido pantalones o chaquetas que son productos muy estratégicos en el surtido. La segunda limitación que se ha encontrado es el sesgo de algunos atributos en la muestra lo cual ha llevado a retener aquellas clases en las variables que explicaban el 90% de la muestra. Ello implica que el modelo obtenido no es generalizable y solo es válido para los 12 países seleccionados, para las 900 marcas, los 50 retailers. Otra limitación es haber utilizado la técnica sub-muestreo para balancear las clases de la variable objetivo lo cual es discutible por la pérdida de información que supone. Asimismo, la modelización se ha hecho más complicada debido a la categorización de la variable target. Finalmente, es oportuno indicar la limitación técnica, se ha requerido un equipo computacional más potente que el doméstico debido al tiempo de procesado.

Las posibilidades de seguir trabajando con este conjunto de datos son varias y distintas gracias a lo que ofrece en la actualidad el aprendizaje automático y el desarrollo computacional que avanza con gran rapidez. Por ello el presente trabajo ofrece la oportunidad de ampliar conocimientos e investigación la línea de investigación de ML. Por un lado, se puede seguir todo el proceso y buscar un algoritmo supervisado tratando la variable objetivo como continua mediante una regresión lineal y algoritmos competidores. Por otro lado, se puede analizar la variable objetivo transformándola a logaritmo para ver si mejorar los resultados. Otra posible vía de investigación es realizar el ejercicio de clasificación de la variable objetivo respecto a los días de supervivencia de la prenda. Por otra parte, haría falta añadir algoritmos de clasificación que aquí no se han utilizado por problemas de rendimiento computacional como es el caso de la red neuronal. Otro trabajo interesante a llevar a cabo es comparar la librería de *scikit-learn* con otras que también se usan para clasificación como el *TensorFlow* que es una biblioteca de código abierto para aprendizaje automático. Por supuesto, someter el conjunto de datos, si bien trataría un objetivo distinto de estudio, al aprendizaje no supervisado aportaría sin duda conocimientos relevantes para la toma de decisiones en una industria tan compleja como la moda.

ANEXOS

Anexo 1. Ventas de ropa y calzado en el sector moda por segmentos

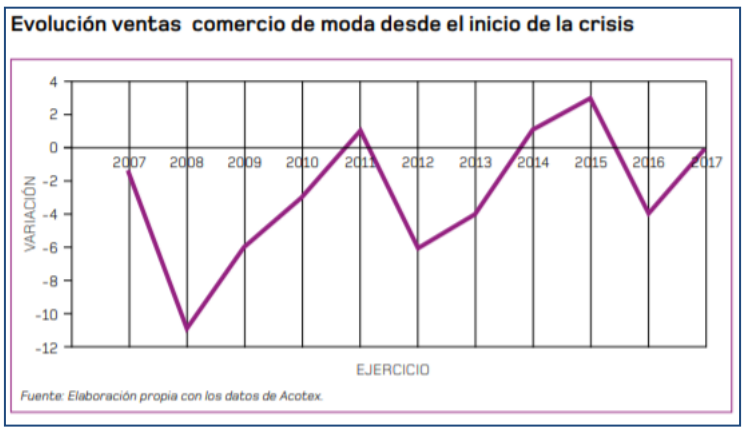


Fuente: Euromonitor internacional

Anexo 2. Países con mayor gasto en ropa

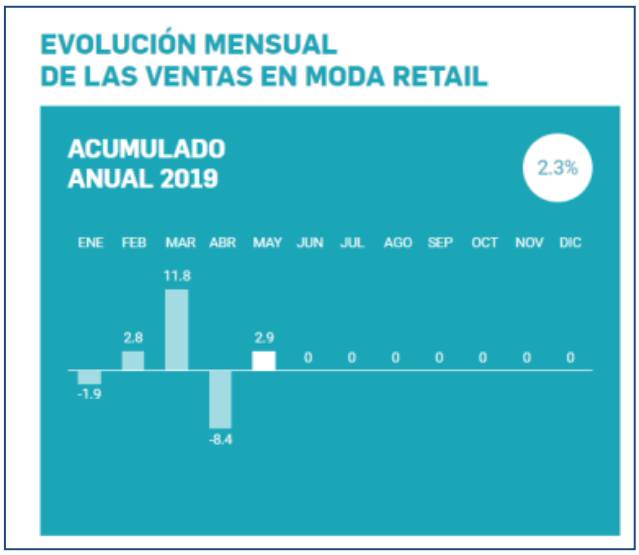
Fuente: http://marketing.eae.es/prensa/SRC_SectorTextil.pdf

Anexo 3. Evolución de las ventas de la industria de la moda



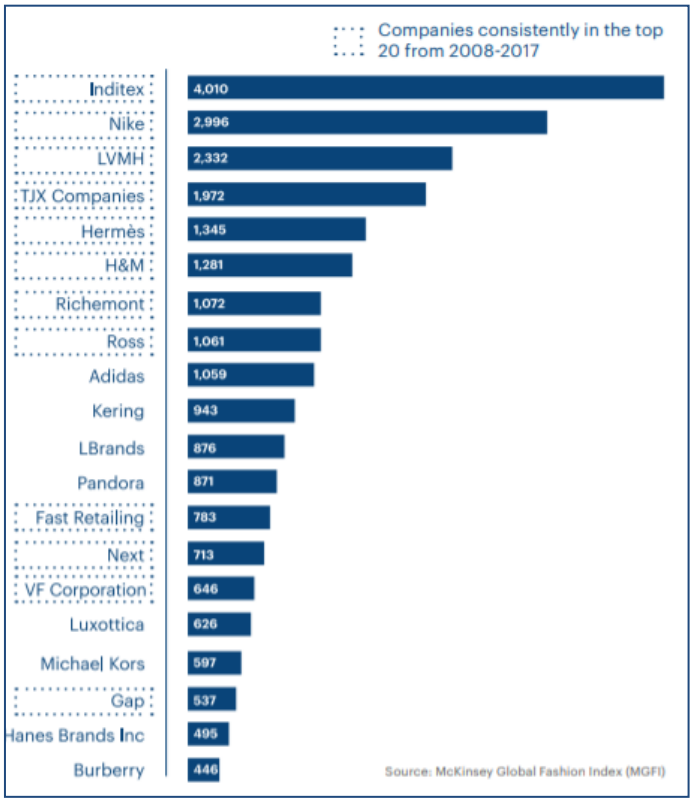
Fuente: https://www.mercasa.es

Anexo 4. Evolución mensual de las ventas en España en moda retail, año 2019



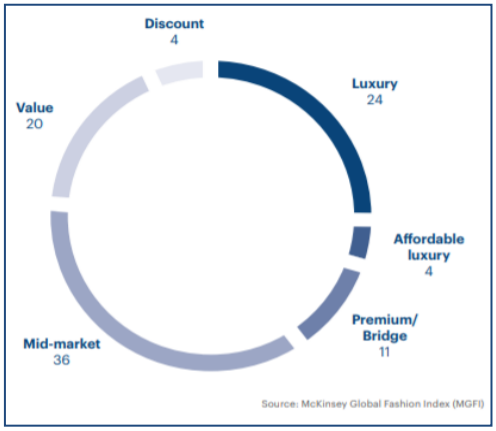
Fuente: https://www.acotex.org/barometro/

Anexo 5. Ranking de retailers de moda en el mundo



Fuente: McKinsey (2019)

Anexo 6. Ventas por segmento en moda



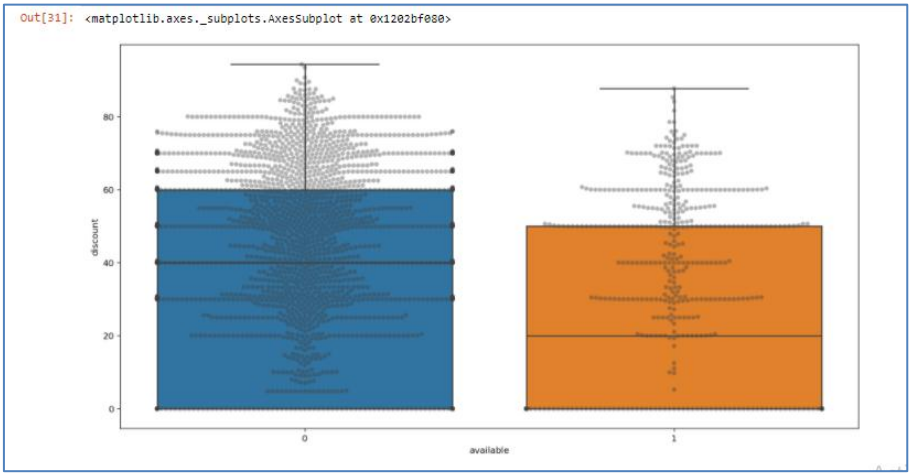
Fuente: McKinsey (2019)

Anexo 7. Análisis bivariante

Atributo available

```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='available', y='discount',data=dfx);
seaborn.swarmplot(x='available', y='discount',data=dfx, color=".25", alpha=0.4)
```

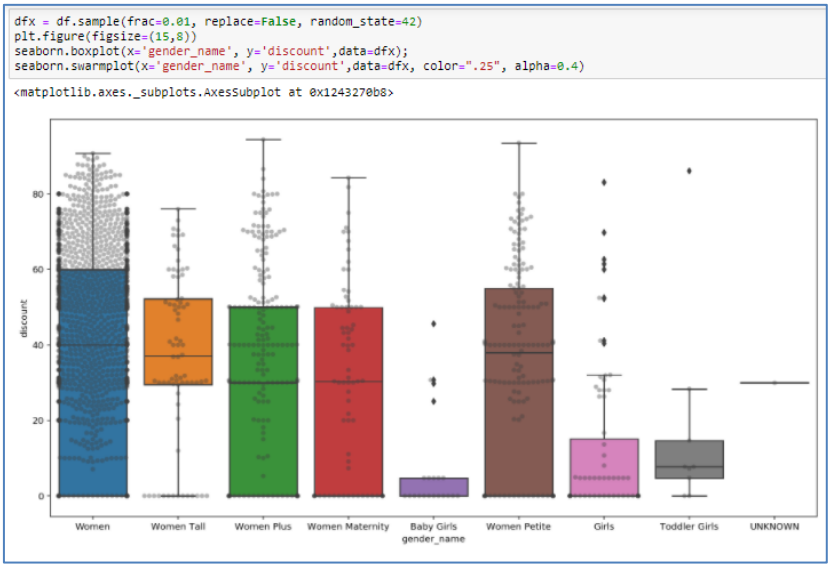
```
<matplotlib.axes._subplots.AxesSubplot at 0x1202bf080>
```



Atributo Gender name

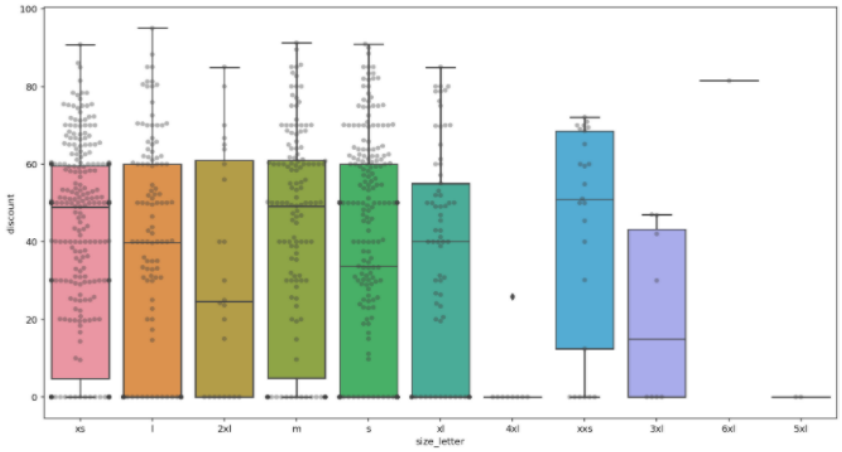
```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='gender_name', y='discount',data=dfx);
seaborn.swarmplot(x='gender_name', y='discount',data=dfx, color=".25", alpha=0.4)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1243270b8>
```



Atributo Size_Letter

```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='size_letter', y='discount',data=dfx);
seaborn.swarmplot(x='size_letter', y='discount',data=dfx, color=".25", alpha=0.4);
```

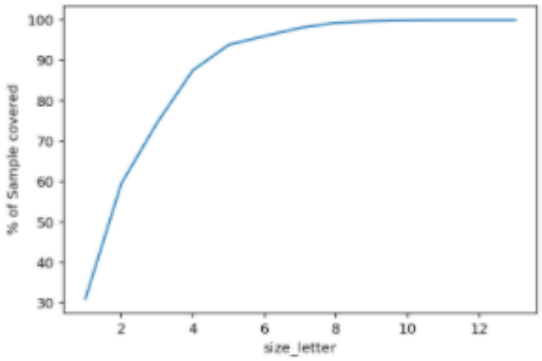



```
x = df[['size_letter', 'discount']].groupby('size_letter').count().reset_index()
x = x.sort_values(by='discount', ascending=False)
x.head()
```

	size_letter	discount
11	xs	31544
9	s	28942
8	m	15332
7	l	13240
10	xl	8402

```
v = list()

for i in numpy.arange(len(x),0,-1):
    val = 100 * x.head(i).discount.sum()/x.discount.sum()
    v.append(val)

seaborn.lineplot(numpy.arange(len(x),0,-1), v)
plt.xlabel('size_letter')
plt.ylabel('% of Sample covered')
```

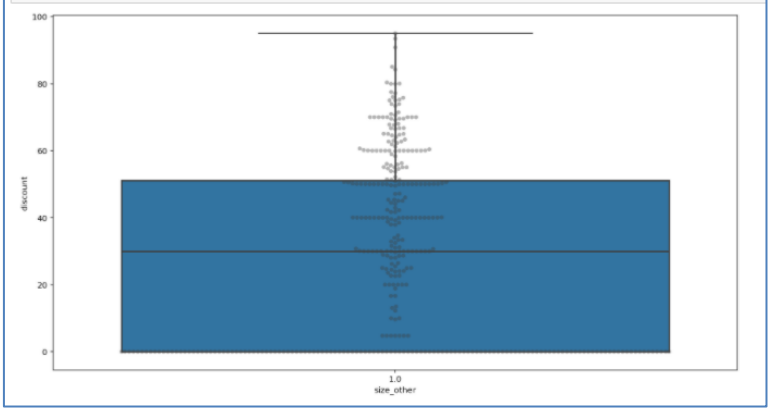
```
Text(0, 0.5, '% of Sample covered')
```



Atributo Size_other

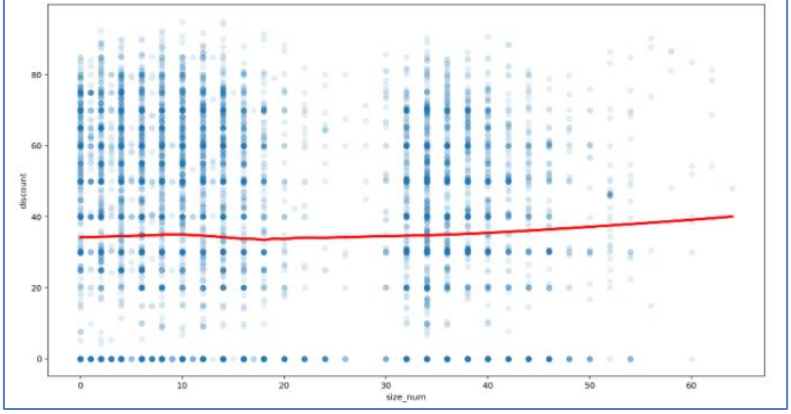
size_other

```
dfx = df.sample(frac=0.01, replace=False, random_state=42)
plt.figure(figsize=(15,8))
seaborn.boxplot(x='size_other', y='discount',data=dfx);
seaborn.swarmplot(x='size_other', y='discount',data=dfx, color=".25", alpha=0.4);
```



Atributo size_num

```
plt.figure(figsize=(15,8))
seaborn.regplot(x='size_num',
                y='discount',
                data=df[~df.size_num.isnull()].sample(frac=0.05, replace=False, random_state=42),
                lowess=True,
                scatter_kws={'alpha':0.1},
                line_kws={'lw':3, 'color':'r'});
```



Anexo 8. Código wordcount para el atributo title

```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\programdata\anaconda3\lib\site-packages (1.5.0)
Requirement already satisfied: numpy>=1.6.1 in c:\programdata\anaconda3\lib\site-packages (from wordcloud) (1.15.4)
Requirement already satisfied: pillow in c:\programdata\anaconda3\lib\site-packages (from wordcloud) (5.3.0)
```

```
# Read in data from pandas
import pandas as pd

# This is used for fast string concatination
from io import StringIO

# Use nltk for valid words
import nltk
import collections as co


import warnings # ignore warnings
warnings.filterwarnings("ignore")
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white", color_codes=True)

# Read the input
d = pd.read_csv("c:/users/belen/desktop/vestido.csv")
```

```
d.head()
```

	PRODUCT_ID	WEBSHOP_NAME	COUNTRY_CODE	COUNTRY_NAME	BRAND	TITLE	DESCRIPTION_TEXT	LANGUAGE_CODE	TRANSLATED_
0	72541521	31 Phillip lim	us	USA	31 Phillip lim	Gown With Gathered Waist	S415-9438CEE-AGEAN-BLUE 100% SILK Model ...	en	
					31	Gown With	S415-9438CEE-MIST 100%		

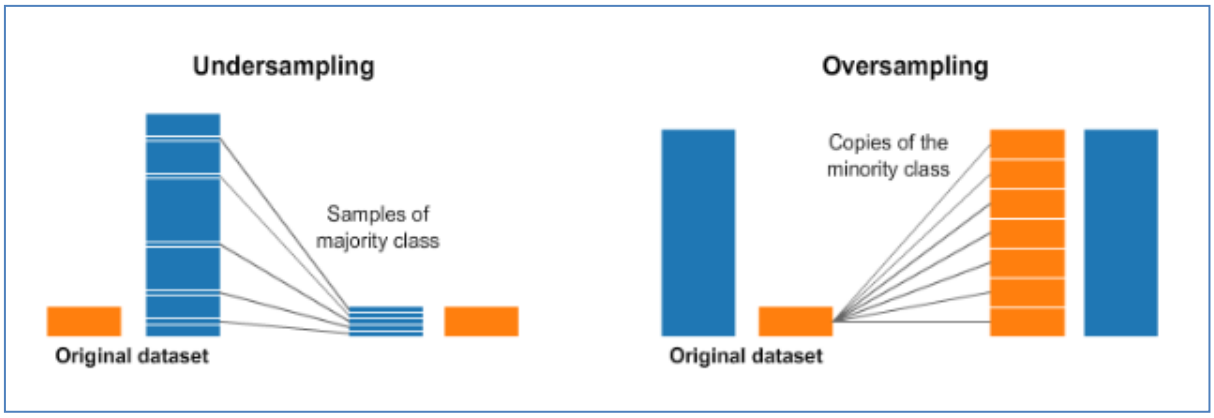
```
si=StringIO()
d['TITLE'].apply(lambda x: si.write(str(x)))
s=si.getvalue()
si.close()
# Note sure how meaningful this is
# but here's a look.
s[0:400]
```

```
'Gown With Gathered WaistGown With Gathered WaistGown With Gathered WaistGown With Gathered WaistGown With Gathered WaistGown W
ith Gathered WaistEmbroidered Gown With Broken Line Back PanelGathered Bodice Gown with Lace InsertsDolman Sleeve Dress with Ga
thered Waist and FringeGathered Bodice Gown with Lace InsertsDolman Sleeve Dress with Gathered Waist and FringeEmbellished-Stra
p GownSunburst-pleat'
```

```
from wordcloud import WordCloud

# Read the whole text.
text = s

# Generate a word cloud image
wordcloud = WordCloud().generate(text)

# Display the generated image:
# the matplotlib way:
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'

# take relative word frequencies into account, lower max_font_size
wordcloud = WordCloud(background_color="white",max_words=len(s),max_font_size=40).generate(text)
plt.figure(figsize=(15,15))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Fuente del código: https://github.com/radzhome/python-word-counter

Anexo 9. Técnicas de Remuestreo



Fuente: https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

Anexo 10. Librerías instaladas e importadas

```
# Install required libraries
dbutils.library.installPyPI('numpy','1.16.3')
# Update python to catch the new versions
dbutils.library.restartPython()

dbutils.library.installPyPI('pandas','0.24.2')
# Update python to catch the new versions
dbutils.library.restartPython()

dbutils.library.installPyPI('scikit-learn','0.20.3')
# Update python to catch the new versions
dbutils.library.restartPython()

dbutils.library.installPyPI('statsmodels','0.9.0')
# Update python to catch the new versions
dbutils.library.restartPython()

dbutils.library.installPyPI('seaborn','0.9.0')
# Update python to catch the new versions
dbutils.library.restartPython()

dbutils.library.installPyPI('matplotlib','3.0.3')
# Update python to catch the new versions
dbutils.library.restartPython()

dbutils.library.installPyPI('xgboost','0.82')
# Update python to catch the new versions
dbutils.library.restartPython()

# Check versions of packages
import pkg_resources
print('numpy ' + pkg_resources.get_distribution('numpy').version)
print('pandas ' + pkg_resources.get_distribution('pandas').version)
print('scikit-learn ' + pkg_resources.get_distribution('scikit-learn').version)
```

```
# Import libraries
import numpy, pandas, sklearn

# Imports
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, LabelEncoder, OrdinalEncoder


from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import f1_score
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
from sklearn.preprocessing import PolynomialFeatures
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import xgboost

import matplotlib as plt
plt.rcParams.update({'figure.max_open_warning': 0})
```

Anexo 11. Preparación del pipeline

```
class CategorySelectorN(BaseEstimator, TransformerMixin):

    def __init__(self, n_cat=10, other_cat_name='other'):
        self.n_cat = n_cat
        self.other_cat_name = other_cat_name
        self.valid_cats = []

    def fit(self, X, y=None):

        if X.shape[1] > 1:
            print('CategorySelectorN cannot handle more than ONE variable')
            raise

        X['target'] = y

        x = X.groupby(X.columns[0]).count().reset_index()
        x = x.sort_values(by='target', ascending=False)

        self.valid_cats = x.head(self.n_cat)[X.columns[0]].values
        X.drop(columns='target', inplace=True)

        return self

    def transform(self, X):

        X.loc[~X[X.columns[0]].isin(self.valid_cats), X.columns[0]] = self.other_cat_name
        return X


class CreateMainMaterial(BaseEstimator, TransformerMixin):

    def __init__(self, keep=True):
        self.material_columns = list()


    def fit(self, X, y=None):
        self.material_columns = [c for c in X.columns if 'material_' in c]
        return self

    def transform(self, X):
        X['main_material'] = X[self.material_columns].idxmax(axis=1).apply(lambda x: x.split('_')[-1])
        return X['main_material'].values.reshape(-1,1)


material_features = [c for c in df.columns if 'material_' in c]
material_transformer = Pipeline(steps=[
    ('mainmaterial', CreateMainMaterial()),
    ('endocer', OrdinalEncoder())
])

webshop_transformer = Pipeline(steps=[
    ('reducer', CategorySelectorN(n_cat=50)),
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OrdinalEncoder())])

country_transformer = Pipeline(steps=[
    ('reducer', CategorySelectorN(n_cat=12)),
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OrdinalEncoder())
])
```

```
brand_transformer = Pipeline(steps=[
    ('reducer', CategorySelectorN(n_cat=100)),
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OrdinalEncoder())
])

size_transformer = Pipeline(steps=[
    ('reducer', CategorySelectorN(n_cat=10)),
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OrdinalEncoder()),
    ])

categoric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OrdinalEncoder())
])

numerical_material_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value=0))
])

numerical_att_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value=0))
])

categorical_att_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value=0))
])



preprocessor_columns = ColumnTransformer(
                transformers=[
                    ('webshop_name', webshop_transformer, ['webshop_name']),
                    ('country_code', country_transformer, ['country_code']),
                    ('brand', brand_transformer, ['brand']),
                    ('main_material', material_transformer, material_features),
                    ('size_letter', size_transformer, ['size_letter']),
                    ('material_percentage', SimpleImputer(strategy='constant',fill_value=0.0), material_features),
                    ('categorical', categoric_transformer, ['gender_name', 'size_letter']),
                    ('numerical_att', numerical_att_transformer, [c for c in df.columns if 'attn_' in c]),
                    ('categorical_att', categorical_att_transformer, [c for c in df.columns if 'att_' in c]),
                    ('numerical_material', numerical_material_transformer, [c for c in df.columns if 'material_' in c]),
                    ('others', 'passthrough', ['available', 'days_alive', 'size_num']),
                    ],
            )

preprocessor = Pipeline(steps=[
    ('preprocessor_columns', preprocessor_columns),
    ('scaler', StandardScaler())
])
```

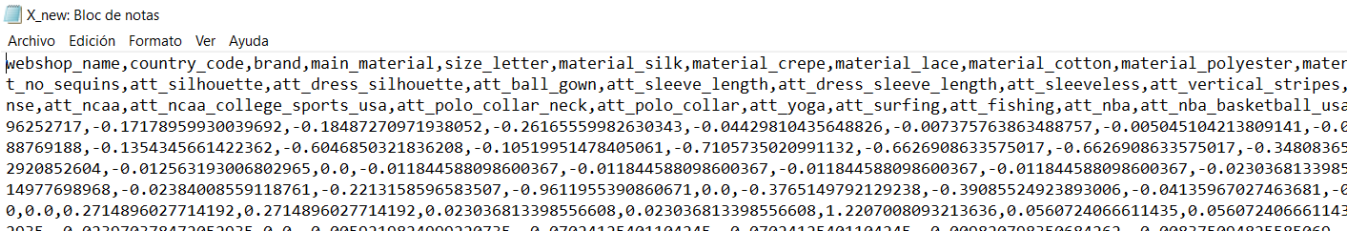
Guardar resultados tras preproceso:

```
X = df.drop(columns=['discount', 'discount_bin'])
le = LabelEncoder()
y =  le.fit_transform(df['discount_bin'])
X_new = preprocessor.fit_transform(X)
```

División del train y test

```
X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.35)
```

Anexo 12. Valores SHAP

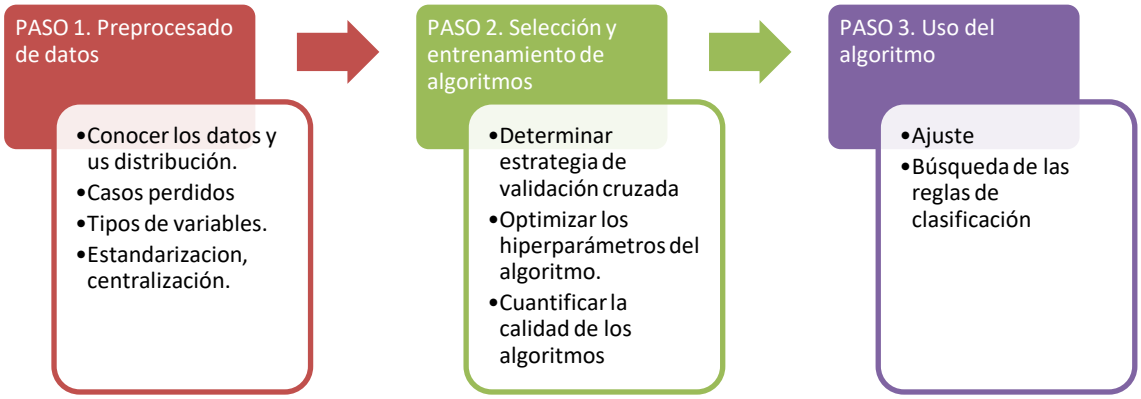

X_new: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
webshop_name,country_code,brand,main_material,size_letter,material_silk,material_crepe,material_lace,material_cotton,material_polyester,mater
t_no_sequins,att_silhouette,att_dress_silhouette,att_ball_gown,att_sleeve_length,att_dress_sleeve_length,att_sleeveless,att_vertical_stripes,
nse,att_ncaa,att_ncaa_college_sports_usa,att_polo_collar_neck,att_polo_collar,att_yoga,att_surfing,att_fishing,att_nba,att_nba_basketball_usa
96252717,-0.17178959930039692,-0.18487270971938052,-0.26165559982630343,-0.04429810435648826,-0.007375763863488757,-0.005045104213809141,-0.0
88769188,-0.1354345661422362,-0.6046850321836208,-0.10519951478405061,-0.7105735020991132,-0.6626908633575017,-0.6626908633575017,-0.3480836
2920852604,-0.012563193006802965,0.0,-0.011844588098600367,-0.011844588098600367,-0.011844588098600367,-0.011844588098600367,-0.0230368133985
14977698968,-0.02384008559118761,-0.2213158596583507,-0.9611955390860671,0.0,-0.3765149792129238,-0.39085524923893006,-0.04135967027463681,-0
0,0.0,0.2714896027714192,0.2714896027714192,0.023036813398556608,0.023036813398556608,1.2207008093213636,0.0560724066611435,0.056072406661143
```

Fuente: conjunto de datos con los valores SHAP para cada muestra

Anexo 13. Etapas del aprendizaje supervisado



Fuente: elaboración propia

BIBLIOGRAFIA

Abawajy, J. (2015). Comprehensive analysis of big data variety landscape. *International journal of parallel, emergent and distributed systems*, 30(1), 5-14.

Acotex (2018). https://www.acotex.org/barometro/. Accedido en abril 2019.

Analytic Insights (2019). https://www.analyticsinsight.net/top-7-big-data-analytics-trends-for-2019/. Recuperado en mayo de 2019.

Aurelie Geron (2019). *Hands-on Machine Learning With Scikit-learn, Keras, And Tensorflow: Concepts, Tools, And Techniques*. Ed. O'Reilly Media

Bandinelli, R., Rinaldi, R., Rossi, M., & Terzi, S. (2013). New product development in the fashion industry: An empirical investigation of Italian firms. *International Journal of Engineering Business Management*, 5(Godište 2013), 5-31.

Barnhart, R. K., & Steinmetz, S. (1988). *The Barnhart dictionary of etymology*. [Bronx, N.Y.]: H.W. Wilson Co.

Barnaghi, P., Sheth, A., & Henson, C. (2013). From data to actionable knowledge: big data challenges in the web of things. *IEEE Intelligent Systems*, (6), 6-11.

Berners-Lee, T., & Shadbolt, N. (2011). There's gold to be mined from all our data. *The Times*, London.

Biau, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, 13(Apr), 1063-1095.

Bruce, M., Daly, L., & Towers, N. (2004). Lean or agile: a solution for supply chain management in the textiles and clothing industry?. *International journal of operations & production management*, 24(2), 151-170.

Brun, A., & Castelli, C. (2013). The nature of luxury: a consumer perspective. International. *Journal of Retail & Distribution Management*, 41(11/12), 823-847.

Cabrera, A., & Frederich, M. (2010). *101 cosas que aprendí en la Escuela de Moda*. Madrid: Abada Editores.

Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS quarterly*, 36(4).

Chen, J., Chen, Y., Du, X., Li, C., Lu, J., Zhao, S., & Zhou, X. (2013). Big data challenge: a data management perspective. *Frontiers of Computer Science*, 7(2), 157-164.

Chen, M., Mao, S., Zhang, Y., & Leung, V. C. (2014). *Big data: related technologies, challenges and future prospects*. Springer.

Choi, T. M. (2017). *Quick response fashion supply chains in the big data Era*. In Optimization and Control for Systems in the Big-Data Era (pp. 253-267). Springer, Cham.

Christopher, M., (2000), The Agile Supply Chain:Competing in Volatile Markets. *Industrial Marketing Management*, Vol 29, pp 37-44.

Christopher, M., Lowson, R., Peck, H., (2004). Creating agile supply chains in the fashion industry. International Journal of Retail & Distribution Management, 32 (8), 367-376.

Corbellini, E., & Saviolo, S. (2012). *Managing Fashion and Luxury Companies*. Florencia. Italia.: Rizzoli Etas.

Cukier, K.(2010). The Economist, Data, data everywhere: A special report on managing information. http://www.economist.com/node/15557443. Accedido en enero de 2019.

Deslandres, Y. (1987). *El traje, imagen del hombre*. Barcelona: Tusquets.

Diebold, F. X. (2012). *A Personal Perspective on the Origin (s) and Development of'Big Data': The Phenomenon, the Term, and the Discipline*. Second Version.

Dillon, S. (2012). *Principios de gestión de empresas de moda*. Barcelona: GustavoGilli.

Dobre, C., & Xhafa, F. (2014). Intelligent services for big data science. *Future Generation Computer Systems*, 37, 267-281.

Domingos, P. M. (2012). A few useful things to know about machine learning. *Commun. acm*, 55(10), 78-87.

EAE (2019). http://marketing.eae.es/prensa/SRC_SectorTextil.pdf. Recuperado en mayo de 2019.

Euromonitor (2019). https://blog.euromonitor.com/global-apparel-footwear-valued-us-1-7-trillion-2017-millions-of-used-clothing-disposed-every-year/. Recuperado en mayo de 2019.

Fernie, J., & Azuma, N. (2004). The changing nature of Japanese fashion: can quick response improve supply chain efficiency?. *European Journal of Marketing*, 38(7), 790-808.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119-139.

Gabarron, L. (1989). *La mística de la moda*. Barcelona: Edit. Anagrama.

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International journal of information management*, 35(2), 137-144.

Gantz, J., & Reinsel, D. (2012). The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, *2007,* 1-16.

Gartner (2019). https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2018/.

Garzan Diccionario (2018). https://www.garzantilinguistica.it. Recuperado en mayo de 2018.

Giunipero, L. C., & Aly Eltantawy, R. (2004). Securing the upstream supply chain: a risk management approach. International *Journal of Physical Distribution & Logistics Management*, 34(9), 698-713.

Grose, V. (2012). *Basics Fashion* Management 1. Suiza: AVA Publising.

Guan C, Guan C, Qin S, Qin S, Ling W, Ling W, Ding G, Ding G. (2016). Apparel recommendation system evolution: an empirical review. *International Journal of Clothing Science and Technology*. Nov 7; 28(6): 854-79.

Hadi Hormozi, Elham Hormozi, and Hamed Rahimi Nohooji. (2012). The Classification of the Applicable Machine Learning Methods in Robot Manipulators. *Int. J. Mach. Learn. Comput.* 2, 5: 560–563.

Harrison, A., Christopher, M. and van Hoek, R. (1999). *Creating the Agile Supply Chain*, Institute of Logistics & Transport, UK.

Hines, T. (2004). The emergence of supply chain management as a critical success factor for retail organisations. *International Retail Marketing*, 108.

IDC(2019). https://www.idc.com/getdoc.jsp?containerId=prUS44998419. Accedido en mayo de 2019.

Jukić, N., Sharma, A., Nestorov, S., & Jukić, B. (2015). Augmenting data warehouses with big data. *Information Systems Management*, 32(3), 200-209.

Kwon, O., Lee, N., & Shin, B. (2014). Data quality management, data usage experience and acquisition intention of big data analytics. *International journal of information management*, 34(3), 387-394.

Labrinidis, A., & Jagadish, H. V. (2012). Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 5(12), 2032-2033.

Laney, D. (2001). *3-D Data Management: Controlling Data Volume, Velocity and Variety.* META Group Research Note, February 6. http://goo.gl/Bo3GS.

Li, T., Zhu, S., & Ogihara, M. (2006). Using discriminant analysis for multi-class classification: an experimental investigation. *Knowledge and information systems*, 10(4), 453-472.

Majeed, A. A., & Rupasinghe, T. D. (2017). Internet of things (IoT) embedded future supply chains for industry 4.0: An assessment from an ERP-based fashion apparel and footwear industry. *International Journal of Supply Chain Management*, 6(1), 25-40.

Malcom, B. (1996). *Fashion as communication*. Nueva York: Routledge.

Mercasa. (2019). https://www.mercasa.es/media/publicaciones/243/1534086521_Moda en_espana_DYC_153_150px.pdf. Recuperado en mayo de 2019.

McAuley, J., Targett, C., Shi, Q., & Van Den Hengel, A. (2015). Image-based recommendations on styles and substitutes. *In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 43-52).

McKinney, W. (2015). pandas: a Python data analysis library de *http://pandas. pydata. org.*

McKinsey (2019). https://www.mckinsey.com/industries/retail/our-insights/the-state-of-fashion-2019-a-year-of-awakening. Recuperado en mayo de 2019.

Ngai, E. W. T., Lam, S. S., Poon, J. K. L., See-To, E. W. K., & To, C. K. M. (2018). *Intelligent Decision Support Prototype System for Fashion Analytics in the Digital Era: An Integrating Visual and Text Approach*.

Observatorio de la Moda Española (2019). http://xn--observatoriomodaespaola-cic.com/wp-content/uploads/2016/06/INFORME.pdf. Recuperado en mayo 2019.

RAE (2018). https://dle.rae.es. Recuperado el 10 de junio de 2018.

Riviere, M (1996). *Diccionario de la moda. Los estilos del siglo XX*. Barcelona: Grijalbo.

Sandhu, R., & Sood, S. K. (2015). Scheduling of big data applications on distributed cloud based on QoS parameters. *Cluster Computing*, 18(2), 817-828.

Saviolo S., Testa S., (2005). *Le imprese del sistema moda - Il management al servizio della creatività*. Etas Libri, Bologna, in Italian.

Şen, A. (2008). The US fashion industry: a supply chain review. *International Journal of Production Economics*, 114(2), 571-593.

Silva, E. S., Hassani, H., Madsen, D. Ø., & Gee, L. (2019). Googling Fashion: Forecasting Fashion Consumer Behaviour Using Google Trends. *Social Sciences,* 8(4), 111.

Sivarajah, U., Kamal, M. M., Irani, Z., & Weerakkody, V. (2017). Critical analysis of Big Data challenges and analytical methods. *Journal of Business Research*, 70, 263-286.

Squicciarino, N. (2012). *El vestido habla: Consideraciones psico-sociológicas sobre la indumentaria*. Madrid: Ediciones Cátedra.

Svensén, M., & Bishop, C. M. (2007). Pattern recognition and machine learning.

Tang, D., Wei, F., Qin, B., Liu, T., & Zhou, M. (2014). Coooolll: A deep learning system for twitter sentiment classification. *In Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014*) (pp. 208-212).

Tech America (2018). https://www.techamerica.org/. Accedido en octubre 2018.

Wang LC, Zeng XY, Koehl L, Chen Y. (2015). Intelligent fashion recommender system: Fuzzy logic in personalized garment design. *IEEE Transactions on Human-Machine Systems*. Feb; 45(1): 95- 109.

Wikipedia (2018). https://es.wikipedia.org/. Recuperado el 10 de junio de 2018.

Yi, X., Liu, F., Liu, J., & Jin, H. (2014). Building a network highway for big data: architecture and challenges. *IEEE Network*, 28(4), 5-13.

Zaslavsky, A., Perera, C., & Georgakopoulos, D. (2013). Sensing as a service and big data. *arXiv preprint arXiv:1301.0159.*

Zhang, Y., Qiu, M., Tsai, C. W., Hassan, M. M., & Alamri, A. (2015). Health-CPS: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal,* 11(1), 88-95.

Zhao, J. L., Fan, S., & Hu, D. (2014). Business challenges and research directions of management analytics in the big data era. *Journal of Management Analytics*, 1(3), 169-174.

INDICE DE ILUSTRACIONES